

VERSION 1.0
DATE – 16/12/2025



PRESENTED BY: AIRPAY PAYMENT SERVICES

AIRPAY PAYMENT SERVICES PVT LTD

GROUND FLOOR GYS INFINITY (PRIUS INFINITY), BAJI PRABHU DESHPANDE MARG, VISHNU PRASAD
SOCIETY, NAVPADA, NETAJI SUBHASH NAGAR, VILE PARLE, MUMBAI, MAHARASHTRA 400057

Android Integration Kit

ABSTRACT

This document specifies the technical aspect of integrating Android SDK.

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	[28/04/2025]	-	[28/04/2025]	First Document
1.1	Tushar Khandekar	[16/12/2025]	-	[16/12/2025]	Minor Changes

Table of Contents

ABSTRACT	2
DISCLAIMER	3
VERSION HISTORY	4
INITIAL CONFIGURATION	6
REQUEST PARAMETER TO CALL AIRPAY SDK –	6
PRIVATEKEY LOGIC	10
CHECKSUM LOGIC.....	10
GRADLE CHANGES	12
RESPONSE MESSAGE	12
JAVA CODE RESPONSE	12
KOTLIN CODE RESPONSE.....	13
RESPONSE FROM THE PAYMENT GATEWAY	16
MANIFEST CHANGES.....	16
SUMMARY	17
FOR SUPPORT –.....	17
REQUEST AND RESPONSE PARAMETER TABLE	18

INITIAL CONFIGURATION

The developer needs to invoke SDK using below code

CODE:

Note:

Set environment variable value as per setup.

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

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

REQUEST PARAMETER TO CALL AIRPAY SDK –

NORMAL TRANSACTION FLOW REQUEST (JAVA CODE)–

```
AirpayConfig.Builder builder = new AirpayConfig.Builder(MainActivity.this, activityResultLauncher);
```

```
builder.setEnvironment(ConfigConstants.PRODUCTION);  
builder.setType(ConfigConstants.AIRPAY_KIT);  
builder.setEncApi(BuildConfig.ENC_API_KEY);  
builder.setPrivateKey(""); /** Generated by Private Key logic  
builder.setMerchantId(""); /** Required  
builder.setOrderId(); /** Required - This should be unique  
builder.setCurrency("356");
```

```
builder.setIsoCurrency("INR");  
builder.setEmailId(); /** Required  
builder.setMobileNo(); /** Required
```

```
builder.setBuyerFirstName("");  
builder.setBuyerLastName("");  
builder.setBuyerAddress("");  
builder.setBuyerCity("");  
builder.setBuyerState("");  
builder.setBuyerCountry("");
```

```
builder.setBuyerPinCode("");
builder.setAmount(); /** Required
builder.setWallet("0");
builder.setCustomVar(""); // Custom message(alphanumeric, space, =)
builder.setTxnSubType(); //In case of Subscction transaction please pass value as "12" and for normal transaction
value as ""
builder.setChmod(""); /*** Required
builder.setChecksum(""); /*** Generated by checksum logic
builder.setMerDom(""); /*** Required
builder.setSuccessUrl(""); /*** Required
builder.setFailedUrl(""); /*** Required
builder.setLanguage("EN");
builder.setClient_id(client_id);
builder.setClient_secret(client_secret);
builder.setGrant_type("client_credentials");
builder.setAesDesKey(sTemp3); // Refer the AesDes Key logic
//In case of Subscription - if txnSubType value as 12
if (!TextUtils.isEmpty(edtTxnSubtype.getText().toString().trim()) &&
edtTxnSubtype.getText().toString().trim().equals("12")) {
builder.setNextrundate(""); /*** Required - In case of subscription
builder.setPeriod(""); /*** Required - In case of subscription
builder.setFrequency(""); /*** Required - In case of subscription
builder.setMaxAmount(""); /*** Required - In case of subscription
builder.setSubscriptionAmt(""); /*** Required - In case of subscription
builder.setIsRecurring(""); /*** Required - In case of subscription
builder.setRecurringCount(""); /*** Required - In case of subscription
builder.setRetryAttempts(""); /*** Required - In case of subscription
builder.build().initiatePayment();
} else {
builder.build().initiatePayment();
}
```

NORMAL TRANSACTION FLOW REQUEST (KOTLIN CODE) –

```
val builder = AirpayConfig.Builder(this@MainActivityKt, activityResultLauncher)
builder.setEnvironment(ConfigConstants.PRODUCTION)
builder.setEncApi(BuildConfig.ENC_API_KEY)
builder.setType(ConfigConstants.AIRPAY_KIT)
builder.setPrivateKey("") /** Generated by Private Key logic
builder.setMerchantId("") /** Required
builder.setOrderId(orderid.editableText.toString().trim { it <= ' ' }) /** Required - This should be unique
builder.setCurrency("356")
builder.setIsoCurrency("INR")
builder.setEmailId("") /** Required
builder.setMobileNo("") /** Required
builder.setBuyerFirstName("")
builder.setBuyerLastName("")
builder.setBuyerAddress("")
builder.setBuyerCity("")
builder.setBuyerState("")
builder.setBuyerCountry("")
builder.setBuyerPinCode("")
builder.setAmount("") /** Required
builder.setWallet("0")
builder.setCustomVar("") // Custom message(alphanumeric, space, =)
builder.setTxnSubType("") //In case of Subscition transaction please pass value as "12" and for normal transaction
value as ""
builder.setChmod("") /*** Required
builder.setChecksum(sChecksum1)
builder.setMerDom("") /*** Required
builder.setSuccessUrl("") /*** Required
builder.setFailedUrl("") /*** Required
builder.setLanguage("EN")
```



```
//In case of Subscription - if txnSubType value as 12
if (!TextUtils.isEmpty(edtTxnSubtype.text.toString().trim { it <= ' ' }) && edtTxnSubtype.text.toString()
.trim { it <= ' ' } == "12") {
builder.setNextrundate("") /** Required - In case of subscription
builder.setPeriod("") /** Required - In case of subscription
builder.setFrequency("") /** Required - In case of subscription
builder.setMaxAmount("") /** Required - In case of subscription
builder.setSubscriptionAmt("") /** Required - In case of subscription
builder.setIsRecurring("1") /** Required - In case of subscription
builder.setRecurringCount("") /** Required - In case of subscription
builder.setRetryAttempts("") /** Required - In case of subscription
builder.build().initiatePayment()
} else
{
builder.build() builder.build().initiatePayment()
}
```

SUBSCRIPTION TRANSACTION FLOW REQUEST –

For subscription transactions, the transaction request must include the **txn sub type** parameter with a value of **12** to indicate a subscription-based payment.

Additionally, the following parameters must be provided:

- Subscription Run Date – Specifies the date when the subscription transaction should be executed.
- Subscription Period – Defines the duration of the subscription (e.g., monthly, yearly).
- Subscription Frequency – Determines how often the transaction should occur within the subscription period.
- Subscription Max Amount – Specifies the maximum allowable amount for the subscription.
- Subscription Amount – Defines the fixed transaction amount for each subscription cycle.
- isRecurring – Indicates whether the transaction is recurring.

- Recurring Count – Specifies the total number of recurring transactions.
- Retry Attempts – Defines the number of times the transaction should be retried in case of failure.

These parameters must be included in the transaction request to enable subscription-based payments.

PRIVATEKEY 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

```
DateFormat df1 = new SimpleDateFormat("yyyy-MM-dd");
String sCurDate1 = df1.format(new Date());
```

```
String sAllData1;
```

(In case of Subscription subtype value is “12” means it is the subscription transaction.
Inside the txnsbtype equals 12 condition contains the subscription sAllSubscriptionData elements.)

```
if (!TextUtils.isEmpty(edtTxnSubtype.getText().toString().trim()) &&
edtTxnSubtype.getText().toString().trim().equals("12")) {
```

```
// In case of Subscription period value is “A” then sAllSubscriptionData contains following elements for checksum
logic
if ("A".equalsIgnoreCase(strPeriod)) {
```

```
sAllSubscriptionData = strPeriod +appendDecimal(edtSubAmt.getText().toString().trim()) +"1" +strRetry;
```

```
} else {
```

```
//sAllSubscriptionData contains following elements if subscription period value is not “A”
```

```
sAllSubscriptionData = edt_sub_next_run_date.getText().toString().trim() +edtFrequency.getText().toString().trim()
+strPeriod +appendDecimal(edtSubAmt.getText().toString().trim()) +"1" +edtRecurringCount.getText().toString() +
strRetry;
```

```
}
```

```
// This is the sAllData1 elements for Subscription Transaction
sAllData1 = emailId.getText().toString().trim() + firstName.getText().toString().trim()
            + lastName.getText().toString().trim() + address.getText().toString().trim()
            + city.getText().toString().trim() + state.getText().toString().trim()
            + country.getText().toString().trim() + appendDecimal(amount.getText().toString().trim())
            + orderid.getText().toString().trim() + sAllSubscriptionData + sCurDate1;

} else {
//This is the sAllData1 elements for Normal Transaction
sAllData1 = emailId.getText().toString().trim() + firstName.getText().toString().trim()
            + lastName.getText().toString().trim() + address.getText().toString().trim()
            + city.getText().toString().trim() + state.getText().toString().trim()
            + country.getText().toString().trim() + appendDecimal(amount.getText().toString().trim())
            + orderid.getText().toString().trim() + sCurDate1;
}

// KEY FOR CHECKSUM -

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

// CHECKSUM –

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

GRADLE CHANGES

Also, add the below line in the build.gradle file –

Please include the following line of code in the default configuration section of the build.gradle file:

buildConfigField "String", "ENC_API_KEY", "" // For value please refer to the sample project shared in the kit

```
implementation 'androidx.multidex:multidex:2.0.0'
implementation("com.airpay:Airpay-India-Kit-V4:1.0.1"){
    exclude group: 'androidx.core', module: 'core'
    // or
    exclude group: 'androidx.legacy', module: 'legacy-support-v4'
}
```

RESPONSE MESSAGE

JAVA CODE RESPONSE

onResult method:

This method provides you with the result of the transaction.

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

        Toast.makeText(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 CONDITON");
}

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");
}
}
}
```


KOTLIN CODE RESPONSE

```
override fun
onResult(o: Any) {
    if (o is Transaction) {
        val transaction = o
        Toast.makeText(this, """"
        ${transaction.transactionstatus}
        ${transaction.statusmsg}
        """"
        .trimIndent(), 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());
}

// Secure Hash Calculation Logic -

val transid = transaction.merchanttransactionid
val apTransactionID = transaction.transactionid
val amount = transaction.transactionamt
val transtatus = transaction.transactionstatus
val message = transaction.statusmsg
var customer_vpa = ""
if (!TextUtils.isEmpty(transaction.chmod) && transaction.chmod.equals("upi", ignoreCase = true)) {
    customer_vpa = ":" + transaction.customervpa
    Log.e("Verified Hash ==", "INSIDE CHMODE UPI CONSIDTION")
}
val merchantid = "" //Please enter Merchant Id
val username = "" //Please enter Username
val sParam =
"$transid:$apTransactionID:$amount:$transtatus:$message:$merchantid:$username$customer_vpa"
val crc = CRC32()
crc.update(sParam.toByteArray())

val sCRC = "" + crc.value
Log.e("Verified Hash ==", "sParam= $sParam")
Log.e("Verified Hash ==", "Calculate Hash= $sCRC")
Log.e("Verified Hash ==", "RESP Secure Hash= " + transaction.securehash)
if (sCRC.equals(transaction.securehash, ignoreCase = true)) {
    Log.e("Verified Hash ==", "SECURE HASH MATCHED")
} else {
    Log.e("Verified Hash ==", "SECURE HASH MIS-MATCHED")
}
}
}
}

```

RESPONSE FROM THE PAYMENT GATEWAY

merchanttransactionid = orderid you have send to airpay system is returned back

transactionid = airpay transaction reference number

transactionamt = transaction amount

status = transaction status (if transaction successful = 200)

statusmsg = Response message received from payment gateway.

securehash = 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.)

MANIFEST CHANGES

In Manifest File, you must mention the following permission -

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

And declare the following SDK class in your Manifest file

```
<activity android:name="com.airpay.airpaysdk_simplifiedotp.AirpayActivity">
```

```
</activity>
```

```
<uses-library
```

```
android:name="org.apache.http.legacy" android:required="false" />
```

SUMMARY

Android is intended to provide an example to Merchant for integrating merchant Payment Solution.

For a summary of features, please see the [Airpay_Android_SDK_Integration.pdf](#) file.

For enabling the UPI intent flow on your merchant app please emailcopying your sales representative.

FOR SUPPORT –

- Tech/integration Support Team
- Customer Support Team

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	Same as Success Url
setCustomVar	String	1-100	Alphanumeric, Space, =
setTxnSubType	String	0-2	<p>Transaction SubType, type of transaction. (length 1-100)</p> <ul style="list-style-type: none"> 1 - INR-auth-capture Allow merchants to first authorize a subscriber's card 2 - INR-sale auth Confirms the cardholder's ability to pay 3 - INR-Moto When a customer makes a card payment over the phone or through mail order 4 - INR-Moto auth-capture When a customer makes a card payment over the phone or through mail order. Allow merchants to first authorize a subscriber's card. 5 - INR-Sale-dcc Dynamic currency conversion 6 - INR-Dcc auth-capture Auth capture in dynamic currency conversion 7 - INR-3 Months 3 Months EMI 8 - INR-6 Months 6 Months EMI 9 - INR-9 Months 9 Months EMI 10 - INR-12 Months 12 Months EMI 11 - INR-18 Months

			18 Months EMI <ul style="list-style-type: none"> 12 - INR-SI Subscription in INR <ul style="list-style-type: none"> 13 - INR-24 Months 24 Months EMI <ul style="list-style-type: none"> 36 - INR-36 Months 36 Months EMI <ul style="list-style-type: none"> 74 - INR-3 Months Debit 3 Months EMI <ul style="list-style-type: none"> 75 - INR-6 Months Debit 6 Months EMI <ul style="list-style-type: none"> 76 - INR-9 Months Debit 9 Months EMI <ul style="list-style-type: none"> 77 - INR-12 Months Debit 12 Months EMI
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 than 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
setEncApi(BuildConfig.ENC_API_KEY)	String	64	Mention enc_api_key in your project of build.gradle file. (For value please refer to the sample project shared in the kit)

)
--	--	--	---

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