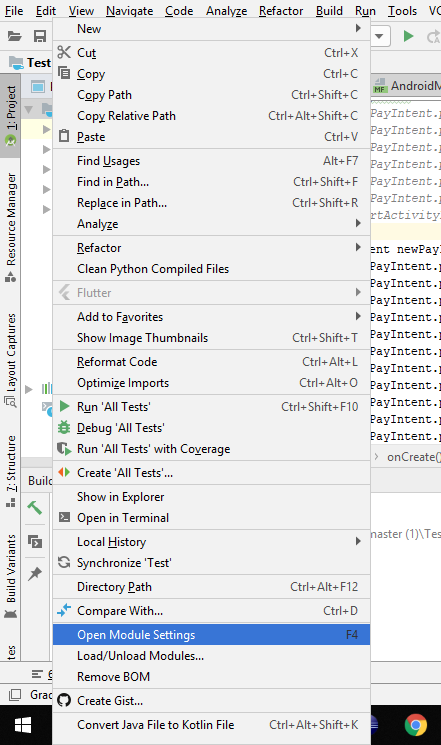
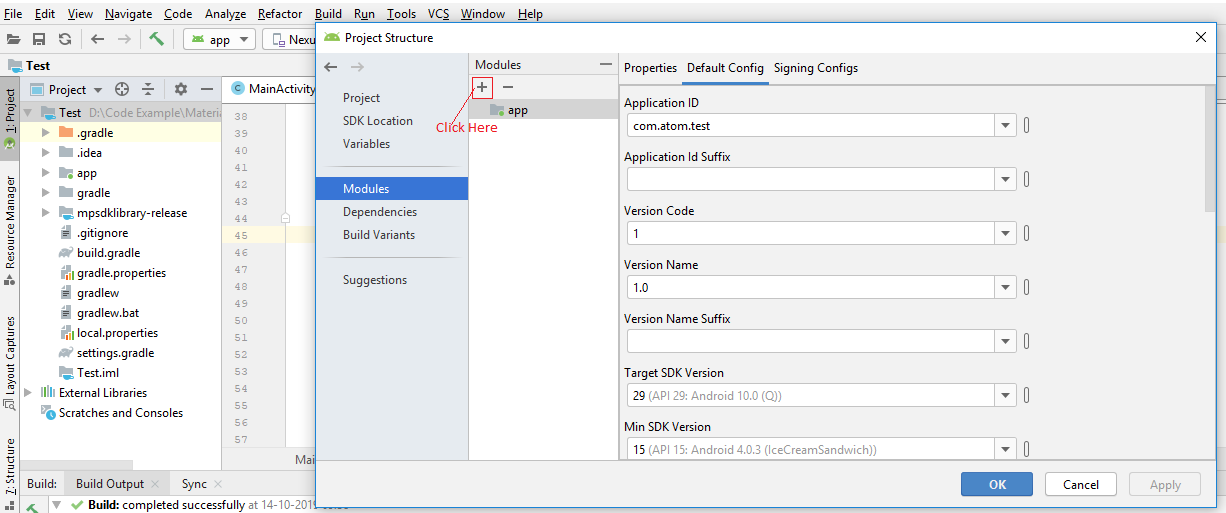
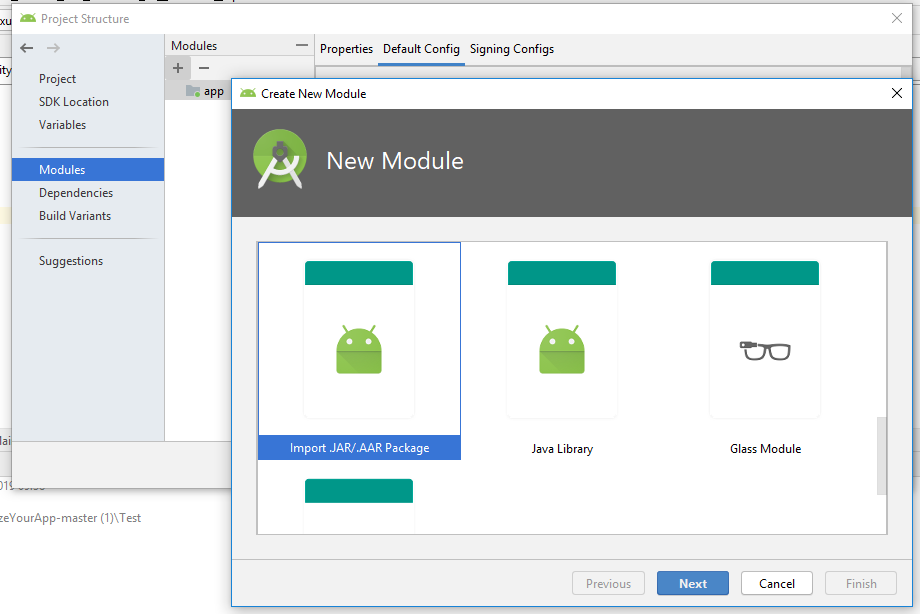
1. Right click on the project and select module setting



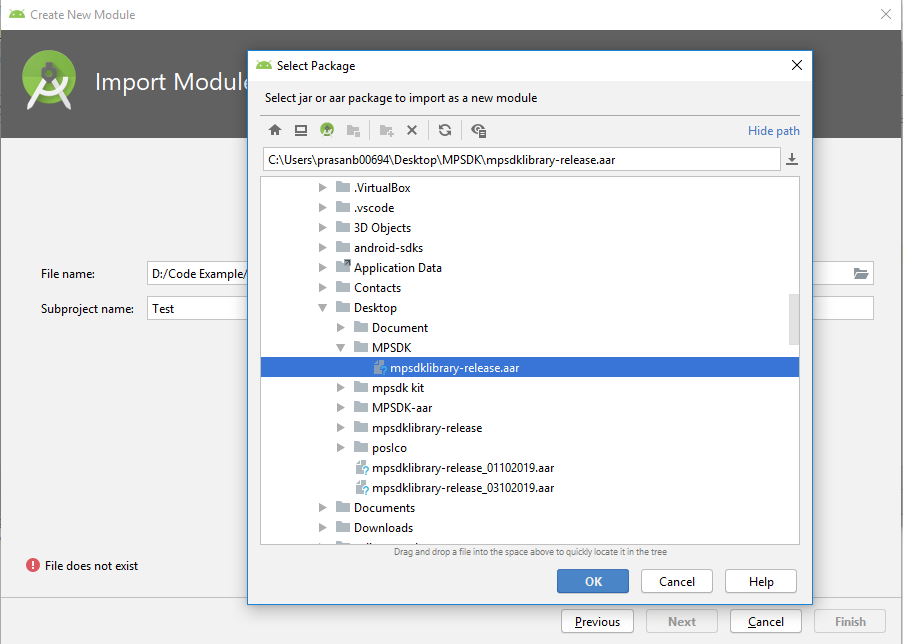
1. Select Modules and click on **+** button to add module



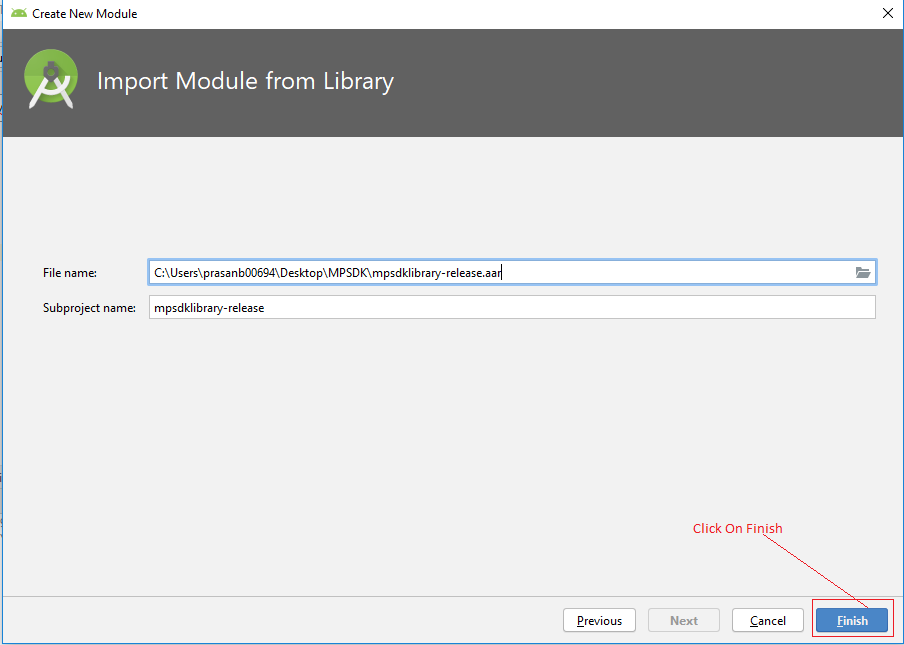
1. On Create New Module window look for the Import .JAR/.AAR Package option and click on Next button.



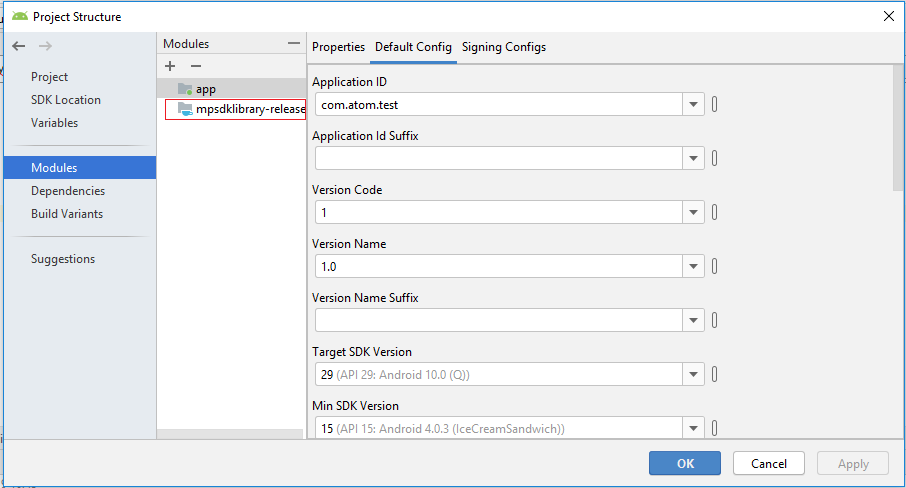
1. Now locate the kit that has been provided by the Atom technologies Limited and select the .aar file from the location and click on OK button.



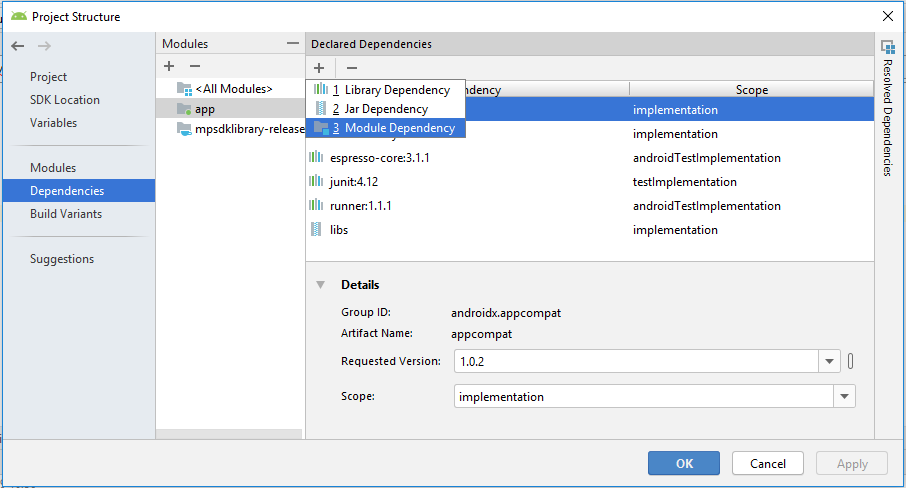
1. Click on Finish button.



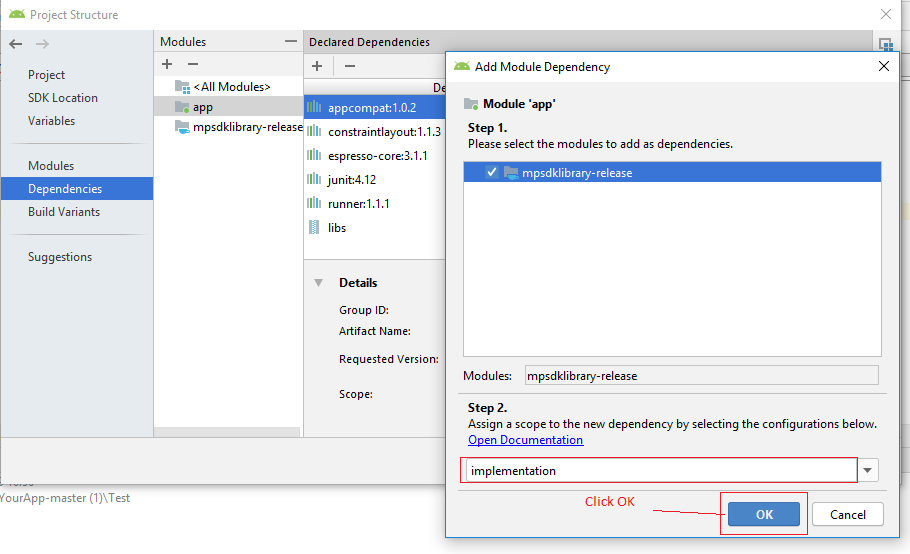
1. Now Library/Kit is added to project



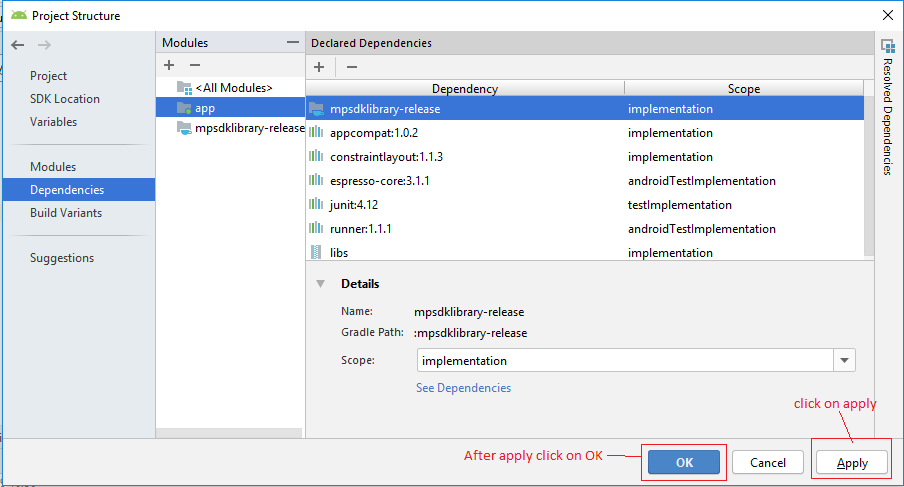
1. Now Click on Dependencies > app > **+** > Module Dependency



1. On Add Module Dependency Window Click on the checkbox to add library file with scope as Implementation and then click on OK button.



1. Once Library dependency is added to the project, click on Apply button and the click on OK Button.



1. To check cross check the integration is complete or not look for the build.gradle(app) file and check for dependencies block and the look for the line

implementation project (**path**: **':mpsdklibrary-release'**)

And check settings.gradle(Project Setting) file and look for the line

**':mpsdklibrary-release'**

1. Add xercesImpl-2.6.2-jaxb-1.0.6.jar file to libs of the project for encoding and add its dependency to build.gradle file of the app.
2. For encoding of clientcode use
3. Use the import statement as

import com.sun.org. apache.xerces.internal.impl.dv.util.Base64;

1. And use the following code to for encoding clientcode

fun encodeBase64(encode: String): String? {  
 println("[encodeBase64] Base64 encode : $encode")  
 var decode: String? = null  
 try {   
 decode = Base64.encode(encode.toByteArray())  
 } catch (e: Exception) {  
 println("Unable to decode : $e")  
 }  
 return decode  
}

1. **mprod (Optional)** : This parameter requires multiple product information that customer/merchant wants to send to the payment gateway, and it should be in xml format. To create xml format use following code and make changes as per requirement.

@Throws(Exception::class)  
 private fun createXmlForProducts(): String?

{

// TODO Auto-generated method stub  
 val factory: DocumentBuilderFactory = DocumentBuilderFactory.newInstance()  
 val builder: DocumentBuilder = factory.newDocumentBuilder()  
 // Here instead of parsing an existing document we want to  
// create a new one.  
 val testDoc: Document = builder.newDocument()  
 // This creates a new tag named 'testElem' inside  
// the document and sets its data to 'TestContent'  
 val lst = ArrayList<String>()  
 lst.add("1,One,250,1,2")  
 lst.add("2,Two,250,1,2,3,4,5")  
   
 var doubleAmt = 0  
 val products: Element = testDoc.createElement("products")  
 testDoc.appendChild(products)  
 for (s in lst) {  
 val line = s.split(",").toTypedArray()  
   
   
 val product: Element = testDoc.createElement("product")  
 products.appendChild(product)  
 val id: Element = testDoc.createElement("id")  
 id.appendChild(testDoc.createTextNode(line[0]))  
 product.appendChild(id)  
 val name: Element = testDoc.createElement("name")  
 name.appendChild(testDoc.createTextNode(line[1]))  
 product.appendChild(name)  
 val amount: Element = testDoc.createElement("amount")  
 amount.appendChild(testDoc.createTextNode(line[2]))  
 product.appendChild(amount)  
 doubleAmt = doubleAmt + line[2].toInt()

amt = Integer.toString(doubleAmt)

if (line.size > 3) {  
 val param1: Element = testDoc.createElement("param1")  
 param1.appendChild(testDoc.createTextNode(line[3]))  
 product.appendChild(param1)  
 }  
 if (line.size > 4) {  
 val param2: Element = testDoc.createElement("param2")  
 param2.appendChild(testDoc.createTextNode(line[4]))  
 product.appendChild(param2)  
 }  
 if (line.size > 5) {  
 val param3: Element = testDoc.createElement("param3")  
 param3.appendChild(testDoc.createTextNode(line[5]))  
 product.appendChild(param3)  
 }  
 if (line.size > 6) {  
 val param4: Element = testDoc.createElement("param4")  
 param4.appendChild(testDoc.createTextNode(line[6]))  
 product.appendChild(param4)  
 }  
 if (line.size > 7) {  
 val param5: Element = testDoc.createElement("param5")  
 param5.appendChild(testDoc.createTextNode(line[7]))  
 product.appendChild(param5)  
 }  
 }

System.out.println("Total Amount :::$amt")

return try {  
 val source = DOMSource(testDoc)  
 val writer = StringWriter()  
 val result = StreamResult(writer)  
 val tf: TransformerFactory = TransformerFactory.newInstance()  
 val transformer: Transformer = tf.newTransformer()  
 transformer.transform(source, result)  
 writer.flush()  
 println ( writer.toString());  
 val s: String = writer.toString().split("\\?").get(0)  
 .substring(1, writer.toString().split("\\?").get(0).length)  
   
 println("Product XML : $s")  
 return s  
 } catch (ex: TransformerException) {  
 ex.printStackTrace()  
 null  
 }  
 }

1. Atom payment gateway for android has provided multiple modes of the payment as follows:
2. **For Net Banking & All Payment Option**:

For Net banking use following snippet of code where you want invoke payment gateway:

val newPayIntent= Intent(this, PayActivity:: class.java);

newPayIntent.putExtra("isLive", false);//true for Live newPayIntent.putExtra("merchantId", "197");//1191  
newPayIntent.putExtra("txnscamt", "0"); //Fixed. Must be �0�  
newPayIntent.putExtra("loginid","197" );  
newPayIntent.putExtra("password", "Test@123");//NCA@1234  
newPayIntent.putExtra("prodid", "NSE");//NCA  
newPayIntent.putExtra("txncurr", "INR"); //Fixed. Must be �INR�  
newPayIntent.putExtra("clientcode",encodeBase64( "007") );  
newPayIntent.putExtra("custacc","100000036600" );  
newPayIntent.putExtra("channelid", "INT");  
newPayIntent.putExtra("amt","100.00" ); //Should be 2 decimal number i.e 1.00  
newPayIntent.putExtra("txnid", "013"); //013  
newPayIntent.putExtra("date", "01/10/2019 18:31:00");//Should be in same format  
newPayIntent.putExtra("signature\_request","KEY123657234" );   
newPayIntent.putExtra("signature\_response","KEYRESP123657234" );  
newPayIntent.putExtra("discriminator","All");//NB,All  
newPayIntent.putExtra("ru","https://paynetzuat.atomtech.in/mobilesdk/param");

//Optinal Parameters  
newPayIntent.putExtra("customerName", "LMN PQR");//Only for Name  
newPayIntent.putExtra("customerEmailID", "Test@gmail.com");//Only for Email ID  
newPayIntent.putExtra("customerMobileNo","8087911057" );//Only for Mobile No.  
newPayIntent.putExtra("billingAddress", "Pune");//Only for Address  
newPayIntent.putExtra("optionalUdf9", "OPTIONAL DATA 2");// Can pass any data  
newPayIntent.putExtra("mprod",mprod );

// Pass data in XML format, only for Multi product  
startActivityForResult(newPayIntent,1);

1. **For CC/DC Payment:**

val newPayIntent= Intent(this, PayActivity:: class.java);  
newPayIntent.putExtra("merchantId", "197");//1191  
newPayIntent.putExtra("txnscamt", "0"); //Fixed. Must be 0  
newPayIntent.putExtra("loginid","197" );  
newPayIntent.putExtra("password", "Test@123");//NCA@1234  
newPayIntent.putExtra("prodid", "NSE");//NCA  
newPayIntent.putExtra("txncurr", "INR"); //Fixed. Must be INR  
newPayIntent.putExtra("clientcode",encodeBase64( "007") );  
newPayIntent.putExtra("custacc","100000036600" );  
newPayIntent.putExtra("channelid", "INT");  
newPayIntent.putExtra("amt","100.00" ); //Should be 2 decimal number i.e 1.00  
newPayIntent.putExtra("txnid", "013"); //013  
newPayIntent.putExtra("date", "01/10/2019 18:31:00");//Should be in same format  
newPayIntent.putExtra("signature\_request","KEY123657234" );   
newPayIntent.putExtra("signature\_response","KEYRESP123657234" );

// CC or DC ONLY (value should be same as commented)  
newPayIntent.putExtra("cardtype", "CC");

// VISA or MASTER or MAESTRO ONLY (value should be same as commented)  
newPayIntent.putExtra("cardAssociate", "VISA");

newPayIntent.putExtra("surcharge", "YES");  
 newPayIntent.putExtra("ru","https://paynetzuat.atomtech.in/mobilesdk/param");

//Optinal Parameters  
newPayIntent.putExtra("customerName", "LMN PQR");//Only for Name  
newPayIntent.putExtra("customerEmailID", "Test@gmail.com");//Only for Email ID  
newPayIntent.putExtra("customerMobileNo","8087911057" );//Only for Mobile No.   
newPayIntent.putExtra("billingAddress", "Pune");//Only for Address  
newPayIntent.putExtra("optionalUdf9", "OPTIONAL DATA 2");// Can pass any data  
// Pass data in XML format, only for Multi product  
newPayIntent.putExtra("mprod",mprod );

startActivityForResult(newPayIntent,1);

1. Add following line of codes in AndroidManifest.xml of the project
2. Add required permissions

<**uses-permission android:name="android.permission.INTERNET"**/>  
<**uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"**/>

1. Add PayActivity to the application tag as

<**activity android:name="com.atom.mpsdklibrary.PayActivity"**/>

1. To get response from the kit use following code where you can use the response and send it /save it to remote /local database :

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  
 super.onActivityResult(requestCode, resultCode, data)  
  
 // check if the request code is same as what is passed here it is 2  
 println("RESULTCODE--->$resultCode")  
 println("REQUESTCODE--->$requestCode")  
 println("RESULT\_OK--->" + Activity.RESULT\_OK)  
  
  
 if (requestCode == 1) {  
 println("---------INSIDE-------")  
 if (data != null) {  
 val message = data.getStringExtra("status")  
 val resKey = data.getStringArrayExtra("responseKeyArray")  
 val resValue = data.getStringArrayExtra("responseValueArray")  
 if (resKey != null && resValue != null) {  
 for (i in resKey.indices)  
 println(" " + i + " resKey : " + resKey[i] + " resValue : " + resValue[i])  
 }  
 Toast.makeText(this, message, Toast.LENGTH\_LONG).show()  
 println("RECEIVED BACK--->$message")  
 }  
 }  
  
}

1. Response ::

//Response from MPSDK kit Android

date :: Thu Oct 31 12:18:01 IST 2019

surcharge :: 0.00

CardNumber :: null

prod :: NSE

clientcode :: 007

mmp\_txn :: 700004570897

signature :: becaaf1e418f8bde738dd2a59004674a9532cea66e36bacd7b4eefe120133848b43ff8c4025f7f741b62a32786ad0d1fbcddd4c677f803e0a986d832e005cc16

udf5 :: null

amt :: 10.00

udf6 :: null

udf3 :: 8087911057

merchant\_id :: 197

udf4 :: Pune

udf1 :: LMN PQR

udf2 :: Test@gmail.com

discriminator :: NB

mer\_txn :: 013

bank\_txn :: 7000045708971

udf9 :: OPTIONAL DATA 2

bank\_name :: Atom Bank

f\_code :: success\_00

1. Requery API is handled at MPSDK side and response of the requery API is

// Response from requery API

MerchantID="197"

MerchantTxnID="013"

AMT="10.00"

VERIFIED="SUCCESS"

BID="7000045708971"

bankname="Atom Bank"

atomtxnId="700004570897"

discriminator="NB"

surcharge="0.00"

CardNumber=""

TxnDate="2019-10-31 12:17:58"

UDF9="OPTIONAL DATA 2" r

econstatus="NRNS"

sdt="null"

1. For Android Version 28 and above use following line of code in AndroidManifest of the app inside application tag

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