# PayDollar Android Mobile SDK

This section explains integration of PayDollar SDK in merchant android mobile application.

## SDK Integration Steps:

### **SDK Requirements**

JAVA SDK Version: 1.8

Android SDK Version: 17 (Minimum), 28 (Target)

### **SDK Configuration**

* Copy **PayDollar.jar** to **libs** folder.
* Add below lines in the dependencies to project’s gradle file:

implementation files(**'libs/PayDollar.jar'**)

Login Function

### **Prepare Login Data**:

Initialize the [LoginData](#_Class_LoginData_properties) class and prepare the login details.

*/\*\* Initialize LoginData \*/*LoginData loginData = **new** LoginData();  
loginData.setMerchantId(**"12345678"**);  
loginData.setUserId(**"userID"**);  
loginData.setPassword(**"password"**);  
loginData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare Login Request:**

Initialize the LoginRequest class and trigger the login function request.

*/\*\* Initialize LoginRequest \*/*LoginRequest loginRequest = **new** LoginRequest(Login.**this**);  
loginRequest.setLoginData(loginData);  
loginRequest.process();

### **Prepare Login Response Handler:**

Initialize a login event handler to capture the login response and result. (Refer to [LoginResult](#_Class_LoginResult_properties))

*/\*\* Initialize LoginResponse \*/*loginRequest.responseHandler(**new** LoginResponse() {  
 @Override  
 **public void** getResponse(LoginResult loginResult) {

*/\*\* Get Result Code \*/* **int** resultCode = loginResult.getResultCode();  
 **if**(resultCode == ***SUCCESS***) {  
 */\*\* Get Params \*/* String currCode = loginResult.getCurrencyCode();  
 String merName = loginResult.getMerchantName();  
 List<PayMethod> payMethodList = loginResult.getPayMethod();  
 String[] channelTypeList = loginResult.getChannelType();  
 String merClass = loginResult.getMerchantClass();  
 **boolean** amexORboolean = loginResult.isAmexOnlineRefund();  
 **boolean** visaORboolean = loginResult.isVisaOnlineRefund();  
 **boolean** masterORboolean = loginResult.isMasterOnlineRefund();  
 **boolean** jcbORboolean = loginResult.isJcbOnlineRefund();  
 **boolean** enableSMSboolean = loginResult.isEnableMPOSMS();  
 **double** rateDbl = loginResult.getRate();  
 **double** fixedDbl = loginResult.getFixed();  
 **boolean** hideSurchargeboolean = loginResult.isHideSurcharge();  
 String partnerlogo = loginResult.getPartnerLogo();  
 String apiId = loginResult.getApiId();  
 String apipassword = loginResult.getApiPassword();  
 *// Do Something if SUCCESS*

} **else if**(resultCode == ***INV\_MERID***){  
 *// Do Something* } **else if**(resultCode == ***INV\_PASSWORD***){  
 *// Do Something* } **else if**(resultCode == ***NO\_USER***){  
 *// Do Something* } **else if**(resultCode == ***CONN\_ERR***){  
 *// Do Something* }  
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
 *// Do Something if input data error* progressDialog.cancel();  
 Toast.*makeText*(Login.**this**, errorResult.getErrCode() + **" - "** + errorResult.getErrMessage(),  
 Toast.***LENGTH\_LONG***).show();  
 }  
});

Payment Function (Scan QR Flow)

### **Prepare Payment Data:**

Initialize the [PayData](#_Class_PayData_properties) class and prepare the payment details.

*/\*\* Initialize PayData \*/*PayData payData = **new** PayData();  
payData.setMerchantId(**"123456"**);  
payData.setAmount(**"10.00"**);  
payData.setCurrCode(EnvBase.Currency.***HKD***);  
payData.setPayment(EnvBase.Payment.***SCAN\_QR***);  
payData.setPayType(EnvBase.PayType.***NORMAL\_PAYMENT***);  
payData.setOrderRef(**"123456"**);  
payData.setpMethod(EnvBase.PayMethod.***ALIPAY***);  
payData.setTxnNo(auth\_code);  
payData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare Payment Request:**

Initialize the PayRequest class and trigger the payment function for **SCAN QR** flow.

*/\*\* Initialize PayRequest \*/*PayRequest payRequest = **new** PayRequest(ScanQRPayment\_2.**this**);  
payRequest.setPayData(payData);  
payRequest.process();

### **Prepare Payment Response Handler:**

Initialize a payment event handler to capture the payment response and result. (Refer to [PayResult](#_Class_PayResult_properties))

*/\*\* Initialize PayResponse \*/*payRequest.responseHandler(**new** PayResponse() {  
 @Override  
 **public void** getResponse(PayResult payResult) {

*/\*\* Get Result Code \*/* **int** resultCode = payResult.getResultCode();  
   
 **if**(resultCode == ***TXN\_SUCCESS***){  
 */\*\* Get Params \*/* **int** prc = payResult.getPrc();  
 **int** src = payResult.getSrc();  
 String merRef = payResult.getMerchantRef();  
 String payRef = payResult.getPayDollarRef();  
 String bankRef = payResult.getBankRef();  
 String amount = payResult.getAmount();  
 String currency = payResult.getCurrency();  
 String payMethod = payResult.getPayMethod();  
 String txnTime = payResult.getTxnTime();  
 String authId = payResult.getAuthId();  
 String bankMerId = payResult.getBankMerId();  
 String bankTerminalId = payResult.getBankTerminalId();  
 *// Do Something if txn is SUCCESS* }**else if**(resultCode == ***TXN\_FAILED***){  
 *// Do Something if txn is FAILED* }  
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
 *// Do Something if input data error* Toast.*makeText*(ScanQRPayment\_2.**this**, errorResult.getErrCode() + **" - "** + errorResult.getErrMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
});

Payment Function (Present QR Flow)

### **Prepare Payment Data:**

Initialize the [PayData](#_Class_PayData_properties) class and prepare the payment details.

*/\*\* Initialize PayData \*/*PayData payData = **new** PayData();  
payData.setMerchantId(**"123456"**);  
payData.setAmount(**"10.00"**);  
payData.setCurrCode(EnvBase.Currency.***MYR***);  
payData.setPayment(EnvBase.Payment.***PRESENT\_QR***);  
payData.setPayType(EnvBase.PayType.***NORMAL\_PAYMENT***);  
payData.setOrderRef(**"123456"**);  
payData.setpMethod(EnvBase.PayMethod.***GRABPAY***);  
payData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare Payment Request:**

Initialize the PayRequest class and trigger the payment function for **PRESENT QR** flow.

*/\*\* Initialize PayRequest \*/*PayRequest payRequest = **new** PayRequest(PresentQRPayment\_3.**this**);  
payRequest.setPayData(payData);  
payRequest.process();

### **Prepare Payment Response Handler:**

Initialize a payment event handler to capture the payment response and result. (Refer to [PayResult](#_Class_PayResult_properties))

*/\*\* Initialize PayResponse \*/*payRequest.responseHandler(**new** PayResponse() {  
 @Override  
 **public void** getResponse(PayResult payResult) {  
  
 */\*\* Get Result Code \*/* **int** resultCode = payResult.getResultCode();  
  
 **if**(resultCode == ***TXN\_SUCCESS***){  
 */\*\* Get Params \*/* **int** prc = payResult.getPrc();  
 **int** src = payResult.getSrc();  
 String merRef = payResult.getMerchantRef();  
 String payRef = payResult.getPayDollarRef();  
 String bankRef = payResult.getBankRef();  
 String amount = payResult.getAmount();  
 String currency = payResult.getCurrency();  
 String payMethod = payResult.getPayMethod();  
 String txnTime = payResult.getTxnTime();  
 String bankMerId = payResult.getBankMerId();  
 String bankTerminalId = payResult.getBankTerminalId();  
 String QRCode = payResult.getQRCode();  
 String QRRef = payResult.getQRRef();  
 String QRType = payResult.getQRCodeType();  
 *// Do Something if transaction is SUCCESS* }**else if**(resultCode == ***TXN\_FAILED***){  
 *// Do Something if transaction is FAILED* }  
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
 Toast.*makeText*(PresentQRPayment\_3.**this**, errorResult.getErrCode() + **" - "** + errorResult.getErrMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
});

Inquiry Payment Function (Present QR Flow)

### **Prepare Inquiry Payment Data:**

Initialize the [InquiryData](#_Class_InquiryData_properties) class and prepare the inquiry payment details.

*/\*\* Initialize InquiryData \*/*InquiryData inquiryData = **new** InquiryData();  
inquiryData.setMerchantId(**"123456"**);  
inquiryData.setPayRef(**"123456"**);  
inquiryData.setpMethod(EnvBase.PayMethod.***GRABPAY***);  
inquiryData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare Inquiry Payment Request:**

Initialize the InquiryRequest class and trigger the inquiry payment function.

*/\*\* Initialize InquiryRequest \*/*InquiryRequest inquiryRequest = **new** InquiryRequest(PresentQRPayment\_3.**this**);  
inquiryRequest.setInquiryData(inquiryData);  
inquiryRequest.process();

### **Prepare Inquiry Payment Response Handler:**

Initialize an inquiry event handler to capture the inquiry response and result. (Refer to [InquiryResult](#_Class_InquiryResult_properties))

*/\*\* Initialize InquiryResponse \*/*inquiryRequest.responseHandler(**new** InquiryResponse() {  
 @Override  
 **public void** getResponse(InquiryResult result) {  
  
 */\*\* Get Result Code \*/* **int** resultCode = result.getResultCode();  
  
 */\*\* Get Params \*/* String returnMsg = result.getReturnMsg();  
 String payRef = result.getPayRef();  
 String bankRef = result.getBankRef();  
 String txnTime = result.getTxnTime();  
  
 **if**(resultCode == InquiryResult.***TXN\_SUCCESS***){  
 *// Do Something if transaction is successful* } **else if**(resultCode == InquiryResult.***TXN\_FAILED***){  
 *// Do Something if transaction is failed* } **else if**(resultCode == ***NOT\_FOUND***){  
 *// Do Something if transaction is not found* } **else if**(resultCode == InquiryResult.***INQUIRY\_FAILED***){  
 *// Do Something if inquiry process is failed* }  
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
 *// Do Something if input data error* Toast.*makeText*(PresentQRPayment\_3.**this**, errorResult.getErrCode() + **" - "** + errorResult.getErrMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
});

Cancel Payment Function (Present QR Flow)

### **Prepare Cancel Payment Data:**

Initialize the [CancelData](#_Class_CancelData_properties) class and prepare the cancel payment details.

*/\*\* Initialize CancelData \*/*CancelData cancelData = **new** CancelData();  
cancelData.setMerchantId(**"123456"**);  
cancelData.setPayRef(**"123456"**);  
cancelData.setpMethod(EnvBase.PayMethod.***GRABPAY***);  
cancelData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare Cancel Payment Request:**

Initialize the CancelRequest class and trigger the cancel payment function.

*/\*\* Initialize CancelRequest \*/*CancelRequest cancelRequest = **new** CancelRequest(PresentQRPayment\_3.**this**);  
cancelRequest.setCancelData(cancelData);  
cancelRequest.process();

### **Prepare Cancel Payment Response Handler:**

Initialize a cancel payment event handler to capture the response and result. (Refer to [CancelResult](#_Class_CancelResult_properties))

*/\*\* Initialize CancelResponse \*/*cancelRequest.responseHandler(**new** CancelResponse() {  
 @Override  
 **public void** getResponse(CancelResult result) {  
  
 */\*\* Get Result Code \*/* **int** resultCode = result.getResultCode();  
  
 */\*\* Get params \*/* String returnMsg = result.getReturnMsg();  
 String payRef = result.getPayRef();  
 String bankRef = result.getBankRef();  
 String txnTime = result.getTxnTime();  
  
 **if**(resultCode == ***CANCEL\_SUCCESS***){  
 *// Do Something if payment is cancelled* } **else if**(resultCode == ***CANCEL\_FAILED***){  
 *// Do Something if payment cannot be canncelled* }  
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
 *// Do Something if input data error* Toast.*makeText*(PresentQRPayment\_3.**this**, errorResult.getErrCode() + **" - "** + errorResult.getErrMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
});

Retrieves Transaction Record(s) Function

### **Prepare History Data**:

Initialize the [HistoryData](#HistoryData) class and prepare the data details.

*/\*\* Initialize HistoryData \*/*HistoryData historyData = **new** HistoryData();  
historyData.setMerchantId(**"123456"**);  
historyData.setApiId(**"apiadmin"**);  
historyData.setApiPassword(**"apipassword"**);  
historyData.setStartDate(**"01022020000000"**);  
historyData.setEndDate(**"01032020235959"**);  
historyData.setSortOrder(EnvBase.SortOrder.***ASC***);  
historyData.setOperatorId(**"admin"**);  
historyData.setOrderStatus(EnvBase.OrderStatus.***ACCEPTED***);  
historyData.setPayRef(**"123456"**);  
historyData.setOrderRef(**"123456"**);  
historyData.setPageNumber(1);  
historyData.setPageRecords(10);  
historyData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare History Request:**

Initialize the HistoryRequest class and trigger the retrieve function request.

*/\*\* Initialize HistoryRequest \*/*HistoryRequest historyRequest = **new** HistoryRequest(History\_List.**this**);  
historyRequest.setHistoryData(historyData);  
historyRequest.process();

### **Prepare History Response Handler:**

Initialize a function event handler to capture the response and result. (Refer to [History Result](#_History_Result_details:))

*/\*\* Initialize HistoryResponse \*/*historyRequest.responseHandler(**new** HistoryResponse() {  
 @Override  
 **public void** getResponse(String result) {  
  
 */\*\* Get Result Code \*/* **int** resultCode = jobj.get(**"resultCode"**).getAsInt();  
  
 */\*\* Get Transaction Record(s) in JSON \*/* JsonObject jobj = **new** Gson().fromJson(result, JsonObject.**class**);  
  
 **if**(resultCode == 0){  
 *// Do Something if SUCCESS* } **else if**(resultCode == 0){  
 *// Do Something if FAILED  
  
 /\*\* Get Error Message \*/* String error = jobj.get(**"error"**).getAsString();  
  
 **if** (error.equalsIgnoreCase(**"Invalid API Login ID"**)) {  
 *// Do Something* } **else if**(error.equalsIgnoreCase(**"Invalid API Login Password"**)){  
 *// Do Something* } **else if**(error.equalsIgnoreCase(**"Connection Error"**)){  
 *// Do Something* }  
 }  
 }

@Override  
 **public void** onError(ErrorResult errorResult) {  
 *// Do Something if input data error* Toast.*makeText*(History\_List.**this**, errorResult.getErrCode() + **" - "** + errorResult.getErrMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
});

Void Transaction Function

### **Prepare Void Transaction Data**:

Initialize the [TxnData](#HistoryData) class and prepare the void transaction data details.

*/\*\* Initialize TxnData \*/*TxnData txnVoidData = **new** TxnData();  
txnVoidData.setMerchantId(**"123456"**);  
txnVoidData.setPayRef(**"123456"**);  
txnVoidData.setApiId(**"apiadmin"**);  
txnVoidData.setApiPassword(**"apipassword"**);  
txnVoidData.setActionType(EnvBase.TxnAction.***VOID***);  
txnVoidData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare Void Transaction Request:**

Initialize the TxnRequest class and trigger void function request.

*/\*\* Initialize TxnRequest \*/*TxnRequest txnVoidRequest = **new** TxnRequest(DialogActivity.**this**);  
txnVoidRequest.setTxnData(txnVoidData);  
txnVoidRequest.process();

### **Prepare Void Response Handler:**

Initialize a void function event handler to capture the void response and result. (Refer to [TxnResult](#_Class_TxnResult_properties))

*/\*\* Initialize TxnResponse \*/*txnVoidRequest.responseHandler(**new** TxnResponse() {  
 @Override  
 **public void** getResponse(TxnResult result) {  
  
 */\*\* Get Result Code \*/* **int** resultCode = result.getResultCode();  
  
 */\*\* Get Return Message \*/* String returnMsg = result.getReturnMsg();  
  
 **if** (resultCode == ***SUCCESS***) {  
 *// Do Something if SUCCESS* } **else** {  
 *// Do Something if FAILED* }  
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
 *// Do Something if input data error* Toast.*makeText*(DialogActivity.**this**, errorResult.getErrCode() + **" - "** + errorResult.getErrMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
});

Refund/Partial Refund Transaction Function

### **Prepare Refund Transaction Data**:

Initialize the [TxnData](#HistoryData) class and prepare the refund transaction data details.

*/\*\* Initialize TxnData \*/*TxnData txnRefundData = **new** TxnData();  
txnRefundData.setMerchantId(**"123456"**);  
txnRefundData.setPayRef(**"123456"**);  
txnRefundData.setApiId(**"apiadmin"**);  
txnRefundData.setApiPassword(**"apipassword"**);  
txnRefundData.setActionType(EnvBase.TxnAction.***REFUND***);  
txnRefundData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare Refund Transaction Request:**

Initialize the TxnRequest class and trigger refund function request.

*/\*\* Initialize TxnRequest \*/*TxnRequest txnRefundRequest = **new** TxnRequest(DialogActivity.**this**);  
txnRefundRequest.setTxnData(txnRefundData);  
txnRefundRequest.process();

### **Prepare Refund Response Handler:**

Initialize a refund function event handler to capture the refund response and result. (Refer to [TxnResult](#_Class_TxnResult_properties))

*/\*\* Initialize TxnResponse \*/*txnRefundRequest.responseHandler(**new** TxnResponse() {  
 @Override  
 **public void** getResponse(TxnResult result) {  
  
 */\*\* Get Result Code \*/* **int** resultCode = result.getResultCode();  
  
 */\*\* Get Return Message \*/* String returnMsg = result.getReturnMsg();  
  
 **if** (resultCode == ***SUCCESS***) {  
 *// Do Something if SUCCESS* } **else** {  
 *// Do Something if FAILED* }  
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
 *// Do Something if input data error* Toast.*makeText*(DialogActivity.**this**, errorResult.getErrCode() + **" - "** + errorResult.getErrMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
});

Retrieves Transaction Settlement Record(s) Function

### **Prepare Settlement Data**:

Initialize the [SettlementData](#_Class_SettlementData_properties) class and prepare the data details.

*/\*\* Initialize SettlementData \*/*SettlementData settlementData = **new** SettlementData();  
settlementData.setMerchantId(**merID**);  
settlementData.setApiId(**"apiuser"**);  
settlementData.setApiPassword(**"api1234"**);  
settlementData.setBatchNo(**"000074"**);  
settlementData.setPayBankId(**"First-Data"**);  
settlementData.setOperatorId(**"admin"**);  
settlementData.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);

### **Prepare Settlement Request:**

Initialize the SettlementRequest class and trigger the retrieve function request.

*/\*\* Initialize SettlementRequest \*/*SettlementRequest settlementRequest = **new** SettlementRequest(Settlement.**this**);  
settlementRequest.setSettlementData(settlementData);  
settlementRequest.process();

### **Prepare Settlement Response Handler:**

Initialize a function event handler to capture the response and result. (Refer to [Settlement Result](#_Settlement_Result_details:))

*/\*\* Initialize SettlementResponse \*/*settlementRequest.responseHandler(**new** SettlementResponse() {  
 @Override  
 **public void** getResponse(String result) {  
  
 */\*\* Get Transaction Record(s) in JSON \*/* JsonObject jobj = **new** Gson().fromJson(result, JsonObject.**class**);  
  
 */\*\* Get Result Code \*/* **int** resultCode = jobj.get(**"resultCode"**).getAsInt();  
  
 **if**(resultCode == 0){  
 *// Do Something if SUCCESS* } **else if**(resultCode == 0){  
 *// Do Something if FAILED  
  
 /\*\* Get Error Message \*/* String error = jobj.get(**"error"**).getAsString();  
  
 **if** (error.equalsIgnoreCase(**"Invalid API Login ID"**)) {  
 *// Do Something* } **else if**(error.equalsIgnoreCase(**"Invalid API Login Password"**)){  
 *// Do Something* } **else if**(error.equalsIgnoreCase(**"Connection Error"**)){  
 *// Do Something* }  
 }  
 }

FDMS Sales Request Transaction

### **Prepare FDMS Sales Request Data**:

Initialize the [FdmsVariable](#FDMSVariable) class and prepare the data details.

*/\*\* Initialize FdmsVariable \*/*FdmsVariable fdmsVariable = **new** FdmsVariable();  
fdmsVariable.setMerRef(**"123456"**);  
fdmsVariable.setAmount(**"10"**);  
fdmsVariable.setMerId(**"123456"**);  
fdmsVariable.setMerName(**"TestingMerchant"**);  
fdmsVariable.setCurrCode(EnvBase.Currency.***HKD***);  
fdmsVariable.setPayType(EnvBase.PayType.***NORMAL\_PAYMENT***);  
fdmsVariable.setPayBankId(EnvBase.PayBankId.***FIRST\_DATA***);  
fdmsVariable.setTerminalId(**"123456"**);  
fdmsVariable.setOperatorId(**"admin"**);  
fdmsVariable.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);  
fdmsVariable.setRequestAction(EnvBase.FDRequest.***SALE***);

### **Prepare FDMS Sales Request:**

Declare FdmsRequest class in global variable and initial it to trigger the sales request.

*/\*\* Global variable \*/***private** FdmsRequest **fdmsRequest** = **null**;

*/\*\* Initialize FdmsRequest \*/***fdmsRequest** = **new** FdmsRequest(EnterAmount.**this**);  
**fdmsRequest**.setFdmsVariables(fdmsVariable);  
**fdmsRequest**.process();

### **Prepare FD Sales Response Handler:**

Capture sales response in onActivityResult() function and retrieve response by initializing a function event handler. (Refer to [FdmsVariable](#FDMSVariable))

@Override  
**protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 */\*\* Response from FD \*/* **fdmsRequest**.saleResponse(requestCode, resultCode, data);  
}

*/\*\* Initialize FdmsResponse \*/***fdmsRequest**.responseHandler(**new** FdmsResponse() {

@Override  
 **public void** getResponse(FdmsVariable result) {

}

@Override  
 **public void** onError(ErrorResult errorResult) {

}  
});

FDMS Void Request Transaction

### **Prepare FDMS Void Request Data**:

Initialize the [FdmsVariable](#FDMSVariable) class and prepare the data details.

*/\*\* Initialize FdmsVariable \*/*FdmsVariable fdmsVariable = **new** FdmsVariable();  
fdmsVariable.setMerId(**"123456"**);  
fdmsVariable.setPayRef(**payRef**);  
fdmsVariable.setUserID(**"apiuser"**);  
fdmsVariable.setPassword(**"api1234"**);  
fdmsVariable.setPayBankId(EnvBase.PayBankId.***FIRST\_DATA***);  
fdmsVariable.setInvoiceNo(**"123456"**);  
fdmsVariable.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);  
fdmsVariable.setRequestAction(EnvBase.FDRequest.***VOID***);

### **Prepare FDMS Void Request:**

Declare FdmsRequest class in global variable and initial it to trigger the void request.

*/\*\* Global variable \*/***private** FdmsRequest **fdmsRequest** = **null**;

*/\*\* Initialize FdmsRequest \*/***fdmsRequest** = **new** FdmsRequest(HistoryDetails.**this**);  
**fdmsRequest**.setFdmsVariables(fdmsVariable);  
**fdmsRequest**.process();

### **Prepare FD Sales Response Handler:**

Capture void response in onActivityResult() function and retrieve response by initializing a function event handler. (Refer to [FdmsVariable](#FDMSVariable))

@Override  
**protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 */\*\* Response from FD \*/* **fdmsRequest**.voidResponse(requestCode, resultCode, data);

}

*/\*\* Initialize FdmsResponse \*/***fdmsRequest**.responseHandler(**new** FdmsResponse() {  
 @Override  
 **public void** getResponse(FdmsVariable result) {  
  
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
  
 }  
});

FDMS Reprint Request

### **Prepare FDMS Reprint Request Data**:

Initialize the [FdmsVariable](#FDMSVariable) class and prepare the data details.

*/\*\* Initialize FdmsVariable \*/*FdmsVariable fdmsVariable = **new** FdmsVariable();  
fdmsVariable.setTraceNo(**"123456"**);  
fdmsVariable.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);  
fdmsVariable.setRequestAction(EnvBase.FDRequest.***REPRINT***);

### **Prepare FDMS Reprint Request:**

Declare FdmsRequest class in global variable and initial it to trigger the reprint request.

*/\*\* Initialize FdmsRequest \*/*FdmsRequest fdmsRequest = **new** FdmsRequest(HistoryDetails.**this**);  
fdmsRequest.setFdmsVariables(fdmsVariable);  
fdmsRequest.process();

FDMS Settlement Request

### **Prepare FDMS Settlement Request Data**:

Initialize the [FdmsVariable](#FDMSVariable) class and prepare the data details.

*/\*\* Initialize FdmsVariable \*/*FdmsVariable fdmsVariable = **new** FdmsVariable();  
fdmsVariable.setMerId(**"123456"**);  
fdmsVariable.setPayGate(EnvBase.PayGate.***PAYDOLLAR***);  
fdmsVariable.setRequestAction(EnvBase.FDRequest.***SETTLEMENT***);

### **Prepare FDMS Settlement Request:**

Declare FdmsRequest class in global variable and initial it to trigger the settlement request.

*/\*\* Global variable \*/***private** FdmsRequest **fdmsRequest** = **null**;

*/\*\* Initialize FdmsRequest \*/***fdmsRequest** = **new** FdmsRequest(SettlementCurrent.**this**);  
**fdmsRequest**.setFdmsVariables(fdmsVariable);  
**fdmsRequest**.process();

### **Prepare FD Sales Settlement Handler:**

Capture settlement response in onActivityResult() function and retrieve response by initializing a function event handler. (Refer to [FdmsVariable](#FDMSVariable))

@Override  
**protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 */\*\* Response from FD \*/* **fdmsRequest**.settlementResponse(requestCode, resultCode, data, **payRefArray**);  
}

*/\*\* Initialize FdmsResponse \*/***fdmsRequest**.responseHandler(**new** FdmsResponse() {  
 @Override  
 **public void** getResponse(FdmsVariable result) {  
   
 }  
  
 @Override  
 **public void** onError(ErrorResult errorResult) {  
  
 }  
});

## Appendix A – Class Data & Result

### **Class LoginData properties details:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameter** | **Data Type** | **Mandatory** | **Expected Value and Description** |
| setMerchantId | String | Yes | ID received after registration on Asiapay merchant portal |
| setUserId | String | Yes | User id or username |
| setPassword | String | Yes | User password |
| setPayGate | EnvBase.PayGate | Yes | Name of payment gateway (Refer to [PayGate](#PayGate)) |

### **Class LoginResult properties details:**

|  |  |  |
| --- | --- | --- |
| **Output Parameters** | **Data Type** | **Description** |
| getResultCode | int | Result code of login function triggered (Refer to [Result Code](#_Result_Code_in)) |
| getMerchantName | String | Name of merchant |
| getCurrencyCode | String | Currency code supported (Refer to [Currency](#Currency)) |
| getReturnMsg | String | Message returned of login function triggered |
| getPayMethod | List<PayMethod> | List of payment methods supported (Refer to [PayMethod](#PayMethod)) |
| getChannelType | String [ ] | List of channel type supported |
| getMerchantClass | String | Class of merchant |
| isAmexOnlineRefund | boolean | If online refund for American Express is supported |
| isVisaOnlineRefund | boolean | If online refund for Visa is supported |
| isMasterOnlineRefund | boolean | If online refund for MasterCard is supported |
| isJcbOnlineRefund | boolean | If online refund for JCB is supported |
| isEnableMPOSMS | boolean | If SMS service is available |
| getRate | double | Rate |
| getFixed | double | Fixed |
| isHideSurcharge | boolean | If surcharge is applicable |
| getPartnerLogo | String | Partner Logo of merchant |
| getApiId | String | Login ID of merchant API |
| getApiPassword | String | Password of merchant API |
| getAddressLine1 | String | Address line 1 |
| getAddressLine2 | String | Address line 2 |
| getAddressLine3 | String | Address line 3 |

### **Class PayData properties details:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input Parameter** | **Data Type** | **Mandatory** | | **Expected Value and Description** |
| **Scan** | **Present** |
| setMerchantId | String | Yes | Yes | ID received after registration on Asiapay merchant portal |
| setAmount | String | Yes | Yes | The total amount of transaction (up to 2 decimal places) |
| setOrderRef | String | Yes | Yes | Transaction reference number given |
| setpMethod | EnvBase.PayMethod | Yes | Yes | Reference of the Activity class which is calling login function |
| setTxnNo | String | Yes | No | Transaction number/QR number |
| setOperatorId | String | No | No | Operator ID who handle transaction |
| setPayment | EnvBase.Payment | Yes | Yes | SCAN\_QR (Refer to [Payment](#Payment)) |
| setCurrCode | EnvBase.Currency | Yes | Yes | Currency of the transaction (Refer to [Currency](#Currency)) |
| setPayType | EnvBase.PayType | Yes | Yes | Payment type (Refer to [PayType](#PayType)) |
| setPayGate | EnvBase.PayGate | Yes | Yes | Name of payment gateway (Refer to [PayGate](#PayGate)) |

### **Class PayResult properties details:**

|  |  |  |
| --- | --- | --- |
| **Output Parameters** | **Data Type** | **Description** |
| getResultCode | int | Result code of payment function triggered (Refer to [Result Code](#_Result_Code_in_1)) |
| getMerchantRef | String | Merchant order reference number |
| getPayDollarRef | String | PayDollar payment reference number |
| getBankRef | String | Reference number provided from bank/payment server |
| getAmount | String | Amount of transaction |
| getCurrency | String | Currency of transaction |
| getPayMethod | String | Payment method of transaction |
| getTxnTime | String | Transaction time |
| getPrc | int | Primary response code (Refer to PRC) |
| getSrc | int | Secondary response code (Refer to SRC) |
| getAuthId | String | Authentication ID (For Scan QR only) |
| getBankMerId | String | Bank merchant ID |
| getBankTerminalId | String | Bank terminal ID |
| getReturnMsg | String | Message returned of payment function triggered |
| getQRCode | String | QR code (For Present QR only) |
| getQRRef | String | QR reference number (For Present QR only) |

### **Class InquiryData properties details:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameter** | **Data Type** | **Mandatory** | **Expected Value and Description** |
| setMerchantId | String | Yes | ID received after registration on Asiapay merchant portal |
| setPayRef | String | Yes | PayDollar payment reference number |
| setpMethod | String | Yes | Payment method of transaction to be inquired |
| setPayGate | EnvBase.PayGate | Yes | Name of payment gateway (Refer to [PayGate](#PayGate)) |

### **Class InquiryResult properties details:**

|  |  |  |
| --- | --- | --- |
| **Output Parameters** | **Data Type** | **Description** |
| getResultCode | int | Result code of inquiry payment function triggered (Refer to [Result Code](#_Result_Code_in_3)) |
| getPayRef | String | PayDollar payment reference number |
| getBankRef | String | Reference number provided from bank/payment server |
| getTxnTime | String | Transaction time |
| getReturnMsg | String | Message returned of login function triggered |

### **Class CancelData properties details:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameter** | **Data Type** | **Mandatory** | **Expected Value and Description** |
| setMerchantId | String | Yes | ID received after registration on Asiapay merchant portal |
| setPayRef | String | Yes | PayDollar payment reference number |
| setpMethod | String | Yes | Payment method of transaction to be cancelled |
| setPayGate | EnvBase.PayGate | Yes | Name of payment gateway (Refer to [PayGate](#PayGate)) |

### **Class CancelResult properties details:**

|  |  |  |
| --- | --- | --- |
| **Output Parameters** | **Data Type** | **Description** |
| getResultCode | int | Result code of cancel payment function triggered (Refer to [Result Code](#_Result_Code_in)) |
| getPayRef | String | PayDollar payment reference number |
| getBankRef | String | Reference number provided from bank/payment server |
| getTxnTime | String | Transaction time |
| getReturnMsg | String | Message returned of login function triggered |

### **Class HistoryData properties details:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameter** | **Data Type** | **Mandatory** | **Expected Value and Description** |
| setMerchantId | String | Yes | ID received after registration on Asiapay merchant portal |
| setApiId | String | Yes | Merchant API user ID |
| setApiPassword | String | Yes | Merchat API user password |
| setStartDate | String | Yes | Start date of records to be retreived |
| setEndDate | String | Yes | End date of records to be retreived |
| setSortOrder | EnvBase.SortOrder | No | Order of records to be retreived (Refer to [SortOrder](#SortOrder)) |
| setOperatorId | String | No | Operator who handled transaction |
| setOrderStatus | EnvBase.OrderStatus | No | Transaction status (Refer to [OrderStatus](#OrderStatus)) |
| setPayRef | String | No | PayDollar payment reference number |
| setOrderRef | String | No | Merchant order reference number |
| setPageNumber | int | No | Page Number (Default will be 1) |
| setPageRecords | int | No | Number of record(s) to be displayed in the page number specified |
| setPayGate | EnvBase.PayGate | Yes | Name of payment gateway (Refer to [PayGate](#PayGate)) |

### **History Result details:**

|  |  |  |
| --- | --- | --- |
| **Result Code** | **Expected Outcome/Error Message** | **Description** |
| 0 | Record(s) returned in JSON format (Refer to [Record](#_Record_Attributes_details:) attributes) | Transaction record(s) retrieved successful |
| -1 | Invalid API Login ID | Invalid merchant API user ID |
| Invalid API Login Password | Invalid merchant API user password |
| Connection Error | Failed connection to server |

### **Class TxnData properties details:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameter** | **Data Type** | **Mandatory** | **Expected Value and Description** |
| setMerchantId | String | Yes | ID received after registration on Asiapay merchant portal |
| setApiId | String | Yes | Merchant API user ID |
| setApiPassword | String | Yes | Merchat API user password |
| setPayRef | String | Yes | PayDollar payment reference number |
| setAmount | String | No | The amount to be refunded (up to 2 decimal places)  \* *Mandatory for partial refund transaction* |
| setActionType | EnvBase.ActionType | Yes | Action to be taken on the transaction (Refer to [TxnAction](#TxnAction)) |
| setPayGate | EnvBase.PayGate | Yes | Name of payment gateway (Refer to [PayGate](#PayGate)) |

### **Class TxnResult properties details:**

|  |  |  |
| --- | --- | --- |
| **Output Parameters** | **Data Type** | **Description** |
| getResultCode | int | Result code of void/refund function triggered (Refer to [Result Code](#_Result_Code_in_2)) |
| getReturnMsg | String | Message returned of void/refund function triggered |

### **Class SettlementData properties details:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Parameter** | **Data Type** | **Mandatory** | **Expected Value and Description** |
| setMerchantId | String | Yes | ID received after registration on Asiapay merchant portal |
| setApiId | String | Yes | Merchant API user ID |
| setApiPassword | String | Yes | Merchat API user password |
| setBatchNo | String | No | Batch number |
| setPayBankId | String | No | Acquire bank ID |
| setOperatorId | String | No | Operator who handled transaction |
| setPayGate | EnvBase.PayGate | Yes | Name of payment gateway (Refer to [PayGate](#PayGate)) |

### **Settlement Result details:**

|  |  |  |
| --- | --- | --- |
| **Result Code** | **Expected Outcome/Error Message** | **Description** |
| 0 | Record(s) returned in JSON format (Refer to [Record](#_Record_Attributes_details:) attributes) | Settlement record(s) retrieved successful |
| -1 | Invalid API Login ID | Invalid merchant API user ID |
| Invalid API Login Password | Invalid merchant API user password |
| Connection Error | Failed connection to server |

### **Record Attributes details:**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| total | Total number of matched transaction record(s) |
| records | JSON array which consists of record(s) |
| orderstatus | Transaction status |
| payref | PayDollar payment reference number |
| bankid | Acquire Bank ID |
| orderdate | Transaction time (DDMMYYYYHHMISS) |
| remark | Transaction remark |
| invoiceNo | Invoice number |
| surcharge | Surcharge amount |
| currency | Currency of transaction |
| amount | Amount of transaction |
| accountno | Transaction number/QR number/Card number |
| batchNo | Batch number |
| merRequestAmt | Total amount of transaction including surcharge |
| cardholder | Transaction payment method |
| traceNo | Trace number |
| settle | Status of settlement (“T” if settled) |
| payMethod | Transaction payment method |
| merref | Merchant order reference number |

### **Class FDMSVariable properties details:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Data Type** | **Mandatory** | **Expected Value and Description** |
| merId | String | Yes | ID received after registration on Asiapay merchant portal |
| merRef | String |  | Merchant reference number |
| amount | String |  | Transaction amount |
| cardNo | String |  | Card number |
| processingCode | String |  | Processing code |
| POSEntryMode | String |  | POS entry mode |
| PANSeqNo | String |  | Primary account number sequence number |
| POSCnditionCode | String |  | POS condition code |
| Track2Data | String |  | Card track2 data |
| EnryptedPIN | String |  | Card PIN |
| EMVICCRelatedData | String |  | EMV related fields data |
| InvoiceRef | String |  | Invoice reference number |
| merName | String |  | Merchant name |
| merRequestAmt | String |  | Merchant request amount |
| surcharge | String |  | Surcharge |
| pMethod | String |  | Payment method |
| cardHolder | String |  | Card holder name |
| epMonth | String |  | Card expired month |
| epYear | String |  | Card expired year |
| CVV2Data | String |  | CVV2 data |
| operatorId | String |  | Operator who handled transaction |
| hideSurcharge | String |  | If surcharge is applicable |
| appCode | String |  |  |
| tc | String |  | Transaction Certificate |
| tsi | String |  | Transworld System Inc. |
| atc | String |  | Application Transaction Counter |
| tvr | String |  | Terminal Verification Result |
| appName | String |  |  |
| aid | String |  | Application Identifier |
| action | String |  |  |
| invoiceNo | String |  | Invoice number |
| RRN | String |  | Retrieval reference number |
| batchNo | String |  | Batch number |
| responseCode | int |  | Response code |
| terminalId | String |  | Terminal ID |
| errMsg | String |  | Error message |
| payRefArray | String |  | Payment references array |
| currCode | EnvBase.Currency |  | Currency of the transaction (Refer to [Currency](#Currency)) |
| payBankId | EnvBase.PayBankId |  | Acquire bank ID |
| payType | EnvBase.PayType |  | Payment type |
| payGate | EnvBase.PayGate |  | Payment gateway |
| requestAction | EnvBase.FDRequest |  | FDMS Request Action |

### **Class ENVBase properties details:**

|  |  |
| --- | --- |
| **Parameters** | **Expected Value and Description** |
| PayGate | SDK supported payment gateway   * **PAYDOLLAR** * **SIAMPAY** * **PESOPAY** |
| PayType | SDK supported payment type   * **NORMAL\_PAYMENT**: Sales payment * **HOLD\_PAYMENT**: Authorize payment |
| PayMethod | SDK supported payment method   * **ALIPAY** * **ALIPAY\_HK** * **BOOST** * **GCASH** * **GRABPAY** * **OEPAY** * **PROMPTPAY** * **UNIONPAY** * **WECHATPAY** * **WECHATPAY\_HK** |
| Payment | SDK supported payment flow   * **SCAN\_QR**: Consumer presented QR payment * **PRESENT\_QR**: Merchant presented QR payment * **CARD**: Card payment |
| Currency | SDK supported currency list  HKD, USD, SGD, RMB, JPY, TWD, AUD, EUR, GBP, CAD, MOP, PHP, THB, MYR, IDR, KRW, BND, NZD, SAR, AED, BRL, INR, TRY, ZAR, VND, DKK, ILS, NOK, RUB, SEK, CHF, ARS, CLP, COP, CZK, EGP, HUF, KZT, LBP, MXN, NGN, PKR, PEN, PLN, QAR, RON, UAH, VEF, LKR, KWD |
| SortOrder | Order of transaction record(s) to be retrieved   * **ASC**: Ascending order * **DESC**: Descending order |
| OrderStatus | Transaction status   * **ACCEPTED** * **REJECTED** * **PENDING** * **REFUNDED** * **PARTIAL\_REFUNDED** * **CANCELLED** * **ACCEPTED\_ADJ** * **ALL** |
| TxnAction | Void/Refund transaction completed   * **VOID** * **REFUND** * **PARTIAL \_REFUND** |

## Appendix B – Result Code

### **Result Code in LoginResult:**

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Value** | **Description** |
| SUCCESS | 0 | Credentials provided are correct |
| INV\_MERID | -1 | Invalid merchant ID provided |
| INV\_PASSWORD | -2 | Invalid password provided |
| NO\_USER | -3 | User is not found in server |
| CONN\_ERR | -4 | Connection Error |

### **Result Code in PayResult:**

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Value** | **Description** |
| TXN\_SUCCESS | 0 | Transaction is successful |
| TXN\_FAILED | -1 | Transaction is failed |

### **Result Code in InquiryResult:**

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Value** | **Description** |
| TXN\_SUCCESS | 0 | Transaction is successful |
| TXN\_FAILED | -1 | Transaction is failed |
| NOT\_FOUND | -2 | Transaction cannot be found |
| INQUIRY\_FAILED | -3 | Inquiry process is failed |

### **Result Code in CancelResult:**

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Value** | **Description** |
| CANCEL\_SUCCESS | 0 | Payment is cancelled successful |
| CANCEL\_FAILED | -1 | Payment cannot be cancelled |

### **Result Code in TxnResult:**

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Value** | **Description** |
| SUCCESS | 0 | Void/Refund is successful |
| FAILED | -1 | Void/Refund is failed |

## Appendix C – Error Code

### **Class ErrorCode values details:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Constant Name** | **Value** | **Description** | **Solution** |
| ERR\_LOGINDATA | -1 | Login data is null or empty | Please add login data (Refer to [Prepare Login Data](#_Prepare_Login_Data:) & [LoginData](#_Class_LoginData_properties)) |
| ERR\_MERID | -2 | Merchant ID is null or empty | Please add merchant ID in the data |
| ERR\_USER | -3 | User login ID is null or empty | Please add user login ID in the data |
| ERR\_PASSWORD | -4 | User login password is null or empty | Please add user login password in the data |
| ERR\_PAYGATE | -5 | PayGate is null or empty | Please add merchant ID in the data |
| ERR\_PAYDATA | -6 | Payment data is null or empty | Please add payment data (Refer to [Prepare Payment Data](#_Prepare_Payment_Data:) & [PayData](#_Class_PayData_properties)) |
| ERR\_AMOUNT | -7 | Amount is null or empty | Please add amount in the data |
| ERR\_CURRCODE | -8 | Currency is null or empty | Please add currency in in the data |
| ERR\_PAYTYPE | -9 | Payment type is null or empty | Please add payment type in the data |
| ERR\_REFNO | -10 | Merchant reference number is null or empty | Please add reference number in the data |
| ERR\_PAYMETHOD | -11 | Payment method is null or empty | Please add payment method in the data |
| ERR\_PAYMENT | -12 | Payment is null or empty | Please add payment in the data |
| ERR\_TXNNO | -13 | Transaction number is null or empty | Please add transaction number in the data |
| ERR\_HISTORYDATA | -14 | History data is null or empty | Please add history data. (Refer to [Prepare History Data](#_Prepare_History_Data:) & [HistoryData](#_Class_HistoryData_properties)) |
| ERR\_APIID | -15 | API user ID is null or empty | Please add API user ID in the data |
| ERR\_APIPASSWORD | -16 | API user password is null or empty | Please add API user password in the data |
| ERR\_STARTDATE | -17 | Inquiry start date is null or empty | Please add inquiry start date in the data |
| ERR\_ENDDATE | -18 | Inquiry end date is null or empty | Please add inquiry end date in the data |
| ERR\_TXNDATA | -19 | Void/Refund transaction data is null or empty | Please add void/refund data (Refer to [Prepare Transaction Data](#_Prepare_Transaction_Data:) & [TxnData](#_Class_TxnData_properties)) |
| ERR\_TXNACTION | -20 | Action type is null or empty | Please add action type in the data |
| ERR\_PAYREFNO | -21 | PayDollar reference number is null or empty | Please add PayDollar reference number in the data. |
| ERR\_INQUIRYDATA | -22 | Inquiry payment data is null or empty | Please add inquiry payment data (Refer to [Prepare Inquiry Payment Data](#_Prepare_Inquiry_Payment) & [InquiryData](#_Class_InquiryData_properties)) |
| ERR\_CANCELDATA | -23 | Cancel payment data is null or empty | Please add cancel payment data (Refer to [Prepare Cancel Payment Data](#_Prepare_Cancel_Payment) & [CancelData](#_Class_CancelData_properties)) |
| ERR\_SETTLEMENTDATA | -24 | Settlement data is null or empty | Please add settlement data (Refer to [Prepare Settlement Data](#_Prepare_Settlement_Data:) & [SettlementData](#_Class_SettlementData_properties)) |
| ERR\_BATCHNO | -25 | Batch number is null or empty | Please add batch number in the data |
| ERR\_PAYBANKID | -26 | Acquire bank ID is null or empty | Please add acquire bank ID in the data |

userID=null,

payRefArray=null,

errMsg=Transaction completed,

action=accepted,

currCode=344,

merId=88148945,

hideSurcharge=,

user=null,

currName=null,

RRN=200327134007,

requestAction=updateTxnAccepted,

PANSeqNo=,

invoiceNo=000075,

EnryptedPIN=,

EMVICCRelatedData=, result=, channel=null,

txnTime=null,

merRequestAmt=,

responseCode=0,

CVV2Data=,

merRef=000006,

POSCondtionCode=,

URL=null,

voidTxnCode=null,

batchNo=000075,

settleTxnCode=null,

payBankId=First-Data,

bankRef=null,

aid=A0000000041010,

payMethod=Master,

tc=872D0615CD703FBF,

returnMsg=Transaction Success,

appCode=134007,

remark=null,

pMethod=VISA,

Track2Data=,

traceNo=000125,

errCode=null,

amount=5.0,

status=null,

InvoiceRef=,

epMonth=03,

surcharge=0,

createTxnCode=0,

cardNo=\*\*\*\*\*\*\*\*\*\*\*\*\*914,

payGate=paydollar,

password=null,

updateTxnCode=null,

tvr=0000008010,

ProcessingCode=,

POSEntryMode=,

tsi=EC00,

appName=Debit MasterCard,

payType=N,

epYear=2023,

terminalId=00001332,

payRef=7705116,

merName=FD Testing,

atc=02A9,

operatorId=, c

ardHolder=,

settle=null}