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')
```

i. Login Function

Prepare Login Data:

Initialize the LoginData 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)

```
/** 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();
        }
});
```

ii. Payment Function (Scan QR Flow)

Prepare Payment Data:

Initialize the PayData 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)

```
/** 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();
    }
});
```

iii. Payment Function (Present QR Flow)

Prepare Payment Data:

Initialize the PayData 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)

```
/** 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();
    }
});
```

iv. Inquiry Payment Function (Present QR Flow)

Prepare Inquiry Payment Data:

Initialize the InquiryData 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)

```
/** 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();
```

```
    }
});
```

v. Cancel Payment Function (Present QR Flow)

Prepare Cancel Payment Data:

Initialize the CancelData 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)

```
/** 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();
    }
});
```

vi. Retrieves Transaction Record(s) Function

Prepare History Data:

Initialize the 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)

```
/** 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();
    }
});
```

vii. Void Function

Prepare Void Transaction Data:

Initialize the TxnData 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)

```
/** 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();
    }
});
```

viii. Refund/Partial Refund Function

Prepare Refund Transaction Data:

Initialize the TxnData 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)

```
/** 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();
    }
});
```

ix. Retrieves Transaction Settlement Record(s) Function

Prepare Settlement Data:

Initialize the SettlementData 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)

```
/** 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
                }
            }
        }

        @Override
        public void onError(ErrorResult errorResult) {
            // Do Something if input data error
             Toast.makeText(Settlement.this, errorResult.getErrCode() + " - " +
errorResult.getErrMessage(), Toast.LENGTH_LONG).show();
        }
    });
}
```

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)

Class LoginResult properties details:

Output Parameters	Data Type	Description
getResultCode	int	Result code of login function triggered (Refer to Result Code)
getMerchantName	String	Name of merchant
getCurrencyCode	String	Currency code supported (Refer to Currency)
getReturnMsg	String	Message returned of login function triggered
getPayMethod	List<PayMethod>	List of payment methods supported (Refer to 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)
setCurrCode	EnvBase.Currency	Yes	Yes	Currency of the transaction (Refer to Currency)
setPayType	EnvBase.PayType	Yes	Yes	Payment type (Refer to PayType)
setPayGate	EnvBase.PayGate	Yes	Yes	Name of payment gateway (Refer to PayGate)

Class PayResult properties details:

Output Parameters	Data Type	Description
getResultCode	int	Result code of payment function triggered (Refer to Result Code)
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)

Class InquiryResult properties details:

Output Parameters	Data Type	Description
getResultCode	int	Result code of inquiry payment function triggered (Refer to Result Code)
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)

Class CancelResult properties details:

Output Parameters	Data Type	Description
getResultCode	int	Result code of cancel payment function triggered (Refer to Result Code)
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)
setOperatorId	String	No	Operator who handled transaction
setOrderStatus	EnvBase.OrderStatus	No	Transaction status (Refer to 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)

History Result details:

Result Code	Expected Outcome/Error Message	Description
0	Record(s) returned in JSON format (Refer to Record 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)
setPayGate	EnvBase.PayGate	Yes	Name of payment gateway (Refer to PayGate)

Class TxnResult properties details:

Output Parameters	Data Type	Description
getResultCode	int	Result code of void/refund function triggered (Refer to Result Code)
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)

Settlement Result details:

Result Code	Expected Outcome/Error Message	Description
0	Record(s) returned in JSON format (Refer to Record 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 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**

	<ul><li>**REFUNDED**</li><li>**PARTIAL_REFUNDED**</li><li>**CANCELLED**</li><li>**ACCEPTED_ADJ**</li><li>**ALL**</li></ul>
TxnAction	Void/Refund transaction completed<ul><li>**VOID**</li><li>**REFUND**</li><li>**PARTIAL _REFUND**</li></ul>

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 & LoginData)
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 & PayData)
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 & HistoryData)
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 & TxnData)
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 & InquiryData)
ERR_CANCELDATA	-23	Cancel payment data is null or empty	Please add cancel payment data (Refer to Prepare Cancel Payment Data & CancelData)
ERR_SETTLEMENTDATA	-24	Settlement data is null or empty	Please add settlement data (Refer to Prepare Settlement Data & SettlementData)
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