# PayDollar PayGate

## Integration Guide (Wallet Direct Connection)

## Version 1.4

(Leave Blank Intentionally)

## Copyright Information

**AsiaPay (HK) Limited**
Unit 1701-05, 17/F, K. Wah Centre
191 Java Road
Hong Kong.
Telephone (852) 2538 8278
Fax: (852) 2545 3898
Web site: http://www.asiapay.com

## Confidentiality and Usage

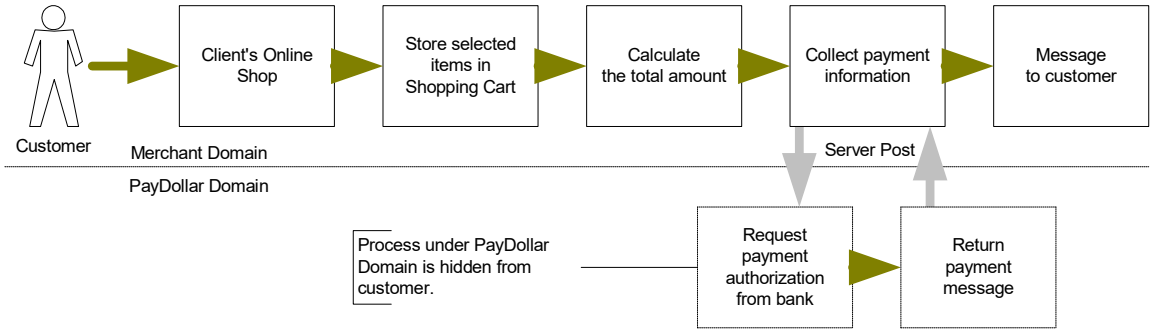
## Revision History

| Revision | Date | Revision Description |
|---|---|---|
| 1.0 | Jul 08, 2020 | First Draft |
| 1.1 | Jul 15, 2020 | Added FPS |
| 1.2 | Aug 11, 2020 | Added OCTOPUS |
| 1.3 | Aug 26, 2020 | Updated Google Pay |
| 1.31 | Sept 07, 2020 | Updated Google Pay Reference Resources |
| 1.4 | Jan 22, 2021 | Added WeChat QR |

# Connection method

## Server Side Direct Connection

This connection method is for merchant to request payment authorization from bank directly through PayDollar PayGate system and subject to approval of acquiring bank. For example, merchant's IVR system or mobile application can directly integrate to us. And in this connection, merchants need to build their own payment information collection page to collect payment information, such as credit card number, expire data, holder's name and etc. Then, payment information has to be sent to a defined URL provided by the acquiring bank. Customer of the merchant, therefore, will not see any bank's payment page.




For merchant who chooses this method of connection, 128-bit SSL cert must be installed for data encryption. The system does not accept non-encrypted data.

PayDollar uses Extended Validation (EV) SSL Certificate. To ensure your system function properly, please check your certificate store can recognize VeriSign intermediate CA certificate - Secure Site Pro/Managed PKI for SSL Premium with EV Certificates. If not, you are required to install the VeriSign intermediate CA certificate in your certificate store.

Please download the primary and secondary VeriSign EV SSL Intermediate CA certificates from the following link then import the 2 certificates into the keystore of your environment.

http://www.verisign.com/support/verisign-intermediate-ca/extended-validation-pro/index.html

(Please be reminded that you should choose the option "Issued After May 17th, 2009")

## Definition of Parameters in the Integration Page

The following are the parameters for integration. PayDollar PayGate is case sensitive. Make sure the typeface is correct. When a transaction is finish, the system will return customer a payment message on the page created by merchant.

| Parameters | Data Type | Descriptions |
|---|---|---|
| **Required Parameter ( with UTF-8 Encoding ) for connect to our payment interface** | | |
| **orderRef** | Text (35) | Merchant's Order Reference Number |
| **amount** | Number (12,2) | Total amount your want to charge the customer [ Up to 2 decimal place ] |
| **currCode** | Text (3) | The currency of the payment:<br>"344" – HKD "840" – USD "702" – SGD<br>"156" – CNY (RMB) "392" – JPY "901" – TWD<br>"036" – AUD "978" – EUR "826" – GBP<br>"124" – CAD "446" – MOP "608" – PHP<br>"764" – THB "458" – MYR "360" – IDR<br>"410" – KRW "682" – SAR "554" – NZD<br>"784" – AED "096" – BND "704" – VND<br>"356" – INR |
| **lang** | Text (1) | The language of the payment page :<br>"E" – English |
| **merchantId** | Number | The merchant ID we provide to you |
| **payType** | Text (1)<br>("N","H") | The payment type:<br>"N" – Normal Payment (Sales)<br>"H" – Hold Payment (Authorize only)<br><br>For merchants who use authorize mode, please be reminded to perform the CAPTURE action as soon as the transaction is confirmed as valid. Once captured, the customer's credit card will be debited in coming bank settlement processing. If the merchant does not capture/reverse the authorized transaction over 14 days, the credit limit will be released to the cardholder after a time period which is subjected to card issuing bank<br><br>Merchant may capture/reverse the authorized transaction in the merchant administration site > Operation > Transaction Detail. |
| **Optional Parameter for billing information** | | |
| **billingFirstName** | Text(60) | First name of customer |
| **billingLastName** | Text(60) | Last name of customer |

| | | |
|---|---|---|
| **billingStreet1** | Text(40) | Address of customer |
| **billingStreet2** | Text(40) | Address of customer ,only mandatory if address exceed 40 |
| **billingCity** | Text(50) | City |
| **billingState** | Text(2) | Mandatory if customer's country is USA or Canada |
| **billingPostalCode** | Text(10) | Mandatory if customer's country is USA or Canada |
| **billingCountry** | Text(2) | Eg.HK |
| **billingEmail** | Text(255) | Email address |
| **custIPAddress** | Text(15) | 192.168.180.100 |
| **Optional Parameter for connect to our payment interface** | | |
| **remark** | Text | An additional remark field that will appear in the confirmation email and transaction detail report to help you to refer the order |
| **secureHash** | Text (40) | Secure hash is used to authenticate the integrity of the transaction information and the identity of the merchant. It is calculated by hashing the combination of various transaction parameters and the Secure Hash Secret.<br><br>*Applies to merchants who registered this function only. For more information, please refer to section 4. |
| **Optional Parameter for using device wallet** | | |
| **eWalletService** | Text<br>("T","F") | eWallet service indicator<br>"T" for device wallet payment<br>"F" for other payment |
| **eWalletBrand** | Text (10) | The value of the device wallet type:<br>"SAMSUNG" – Samsung Pay "APPLEPAY" – Apple Pay<br>"GOOGLE" – Google Pay™ |
| **eWalletPaymentData** | Text | Retrieved device wallet payment data from related SDK with Base64 Format |
| **Optional Parameter for using third party wallet** | | |
| **pMethod** | Text | The payment method type<br>"ALIPAYAPP" – Alipay Global App<br>"ALIPAYCNAPP" – Alipay China App<br>"ALIPAYHKAPP" – Alipay HK App<br>"WECHATAPP" – WeChat Pay App<br>"WECHATONL" – WeChat Pay QR<br>"FPS" – FPS App 2 App<br>"OCTOPUS" – Octopus App 2 App |

| Return Parameter | | |
|---|---|---|
| src | Number | Return bank host status code |
| prc | Number | Return bank host status code |
| Ord | Number | Bank Reference Number |
| Holder | Text | The Holder Name of the Payment Account |
| successcode | Number | Transaction Status:<br>0 – Transaction succeeded<br>1 – Transaction Failure<br>2 – Transaction Pending (for App–2–App integration) |
| Ref | Text | Merchant's Order Reference Number |
| PayRef | Number | PayDollar Payment Reference Number |
| Amt | Number (12,2) | Transaction Amount |
| Cur | Number (3) | Transaction Currency i.e. "344" - HKD |
| AuthId | Text | Approval Code |
| TxTime | Text<br>(YYYY-MM-DD<br>HH:MI:SS.0) | Transaction Time |
| errMsg | Text | Return Message or Data from payment method |

All the return parameters will be concatenated as in html request format by separate with **&**.

Sample return string:

**successcode=0&Ref=Test&PayRef=4780&Amt=1.0&Cur=344&prc=0&src=0&Ord=6697090&Holder=edward&AuthId=123456&TxTime=2003-10-07 17:48:02.0&errMsg=Transaction completed**

## Kick Off

After the integration has been completed, it is ready to launch your e-commerce web to serve your customers. Please copy the following **TESTING URL** for Direct Connect Server Post method:

---

**https://test.paydollar.com/b2cDemo/eng/directPay/payComp.jsp**

---

Please copy the following **PRODUCTION URL** for Direct Connect Server Post method:

---

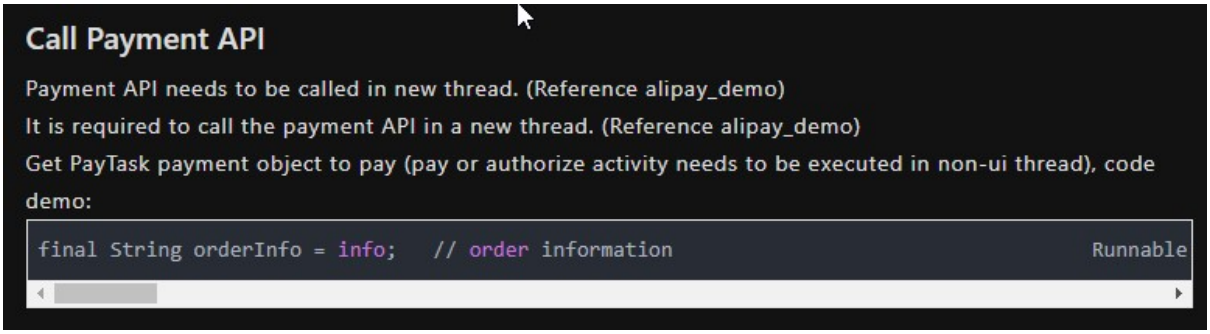
**https://www.paydollar.com/b2c2/eng/directPay/payComp.jsp**

---

# SDK Integration Step

## Alipay Global App
## Alipay China App
## Alipay HK App

1. Send the related order information to PayDollar Server Side Direct Connection.
2. PayDollar will return related transaction information at parameter "**errMsg**".
3. Set the content at "**errMsg**" to the Alipay SDK "order information" string.




4. Initialize Alipay SDK and handle the response.
5. PayDollar will send the data feed to merchant server for the final transaction result.

Related SDK document:

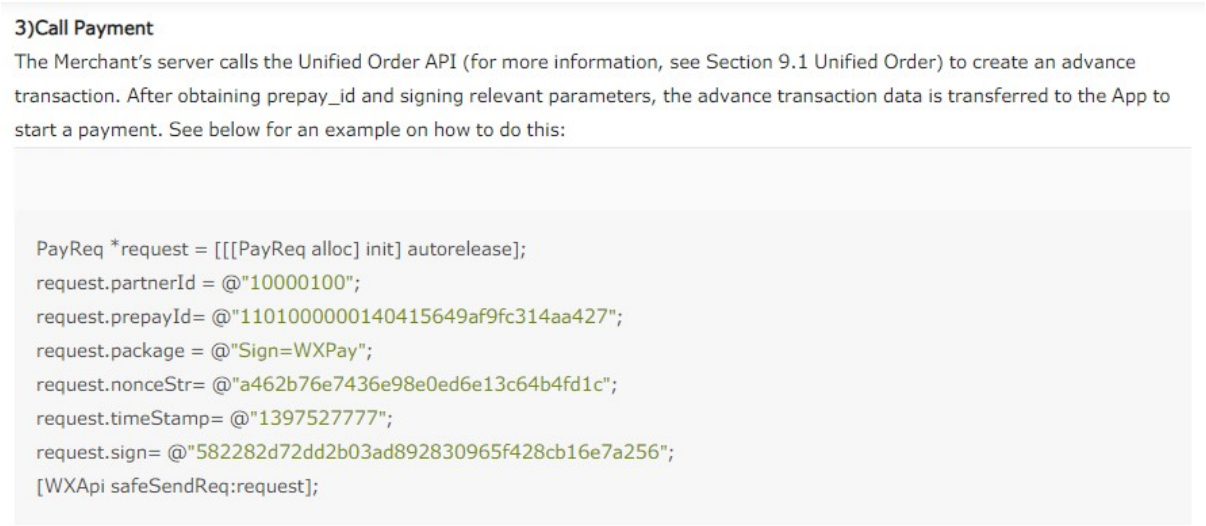
Alipay Global: https://global.alipay.com/docs/ac/app/sdk_integration

Alipay HK: https://global.alipay.com/docs/ac/app_hk/sdkintegration

Alipay China: https://opendocs.alipay.com/open/204/105296

## WeChat Pay App

1. Send the related order information to PayDollar Server Side Direct Connection.
2. PayDollar will return related transaction information at parameter "**errMsg**".
   successful "**errMsg**" format:
   appId|nonceStr|package|partnerId|prepayId|timeStamp|sign|end
3. Split the "**errMsg**" with using "|" and put the data to WeChat SDK with the corresponding variable.



**3)Call Payment**

The Merchant's server calls the Unified Order API (for more information, see Section 9.1 Unified Order) to create an advance transaction. After obtaining prepay_id and signing relevant parameters, the advance transaction data is transferred to the App to start a payment. See below for an example on how to do this:

```
PayReq *request = [[[PayReq alloc] init] autorelease];
request.partnerId = @"10000100";
request.prepayId= @"1101000000140415649af9fc314aa427";
request.package = @"Sign=WXPay";
request.nonceStr= @"a462b76e7436e98e0ed6e13c64b4fd1c";
request.timeStamp= @"1397527777";
request.sign= @"582282d72dd2b03ad892830965f428cb16e7a256";
[WXApi safeSendReq:request];
```

4. Submit the SDK request.
5. PayDollar will send the data feed to merchant server for the final transaction result.

Related SDK document:
https://pay.weixin.qq.com/wiki/doc/api/wxpay/pay/In-AppPay/chapter6_2.shtml

## WeChat Pay QR

1. Send the related order information to PayDollar Server Side Direct Connection with using parameter **pMethod** with value "**WECHATONL**"
2. PayDollar will return the WeChat Pay QR payment token at parameter "**errMsg**" with base64 format.
3. Merchant decode the "**errMsg**" with base64 and convert the payment token to QR image.
4. Merchant display the QR and allow client to send to complete the payment.
5. Merchant can listen the payment result with using Order API Query function or wait the datafeed response at the merchant backend server
6. After payment complete, PayDollar will send the data feed to merchant server for the final transaction result.

Related SDK document:
https://pay.weixin.qq.com/wiki/doc/api/wxpay/en/pay/NativePay/chapter4_1.shtml

## Google Pay™

By using PayDollar Google Pay™ service, please use the below configuration at the Google Pay™ SDK configuration.

- "**gateway**": "asiapay"
- "**gatewayMerchantId**": "PayDollar Merchant Id"

For the below configuration, please contact the PayDollar support team about it. It is related to the acquirer bank and your account supported.

Android:

- getAllowedCardNetworks
- getAllowedCardAuthMethods

Web:

- allowedAuthMethods
- allowedCardNetworks

For the detail, please reference to the following URL:

https://developers.google.com/pay/api/android/reference/request-objects#PaymentMethod

Step:

1. Follow the Google Pay™ SDK integration guide with using PayDollar provided configuration values
2. Implement Google Pay™ and prepare the payment token
   a. Android:

   At the Google Pay™ SDK function **handlePaymentSuccess**

```
/**
 * PaymentData response object contains the payment information, as well as any additional
 * requested information, such as billing and shipping address.
 *
 * @param paymentData A response object returned by Google after a payer approves payment.
 * @see <a href="https://developers.google.com/pay/api/android/reference/
 * object#PaymentData">PaymentData</a>
 */
private void handlePaymentSuccess(PaymentData paymentData) {

  // Token will be null if PaymentDataRequest was not constructed using fromJson(String).
  final String paymentInfo = paymentData.toJson();
  if (paymentInfo == null) {
    return;
  }
```

   Encode the **paymentInfo** with Base64 format

```
byte[] byteString=paymentInfo.getBytes("UTF-8");
String base64encodedString= android.util.Base64.encodeToString(byteString, Base64.NO_WRAP);
```

   b. Web:

   Follow the Google Pay™ JavaScript integration guide and retrieve the paymentToken at **processPayment** function.

```
/**
 * Process payment data returned by the Google Pay API
 *
 * @param {object} paymentData response from Google Pay API after user approves payment
 * @see {@link https://developers.google.com/pay/api/web/reference/response-objects#PaymentData|Pay
 */
function processPayment(paymentData) {
  // show returned data in developer console for debugging
    console.log(paymentData);
  // @todo pass payment token to your gateway to process payment
  paymentToken = paymentData.paymentMethodData.tokenizationData.token;
}</script>
<script async
  src="https://pay.google.com/gp/p/js/pay.js"
  onload="onGooglePayLoaded()"></script>
```

Send the paymentToken to the merchant server and encode it with Base64 format

```
byte[] byteString=paymentToken.getBytes("UTF-8");
String base64encodedString= java.util.Base64.getEncoder().encodeToString(byteString);
```

3. Submit the payment request to PayDollar Server Side Direct Connection with the following parameter:

   **eWalletPaymentData**: Encoded Base64 payment token String

   **eWalletService**: T

   **eWalletBrand**: GOOGLE

4. PayDollar will send the data feed to merchant server for the final transaction result.

Please read and accept the following policy for Google Pay™:

Google Pay™ API Acceptable Use Policy:

https://payments.developers.google.com/terms/aup

Google Pay™ API Terms of Service:

https://payments.developers.google.com/terms/sellertos

Related SDK document:

Android: https://developers.google.com/pay/api/android/overview

Web: https://developers.google.com/pay/api/web/overview

## Apple Pay

1. Follow the Apple account configuration step to create Apple merchant ID and upload the acquiring bank certificate to the account.
2. Follow the Apple Pay SDK integration step and get the Apple Pay token data (**PKPayment**)

> The actions on your server vary depending on whether you process your own payments or work with a payment platform. In both cases, your server handles the order and sends a status back to the device, which your delegate passes to its completion handler, as described in Processing Payments.

```
- (void) paymentAuthorizationViewController:(PKPaymentAuthorizationViewController
*)controller
                       didAuthorizePayment:(PKPayment *)payment
                                completion:(void (^)
(PKPaymentAuthorizationStatus))completion
{
    NSError *error;
    ABMultiValueRef addressMultiValue = ABRecordCopyValue(payment.billingAddress,
kABPersonAddressProperty);
    NSDictionary *addressDictionary = (__bridge_transfer NSDictionary *)
ABMultiValueCopyValueAtIndex(addressMultiValue, 0);
    NSData *json = [NSJSONSerialization dataWithJSONObject:addressDictionary
options:NSJSONWritingPrettyPrinted error: &error];

    // ... Send payment token, shipping and billing address, and order information to your
server ...

    PKPaymentAuthorizationStatus status;  // From your server
    completion(status);
}
```

3. Convert the PKPayment object to JSON format and encode it with **Base64** format

Example JSON for PKPayment:

```
{
  "token":{
    "paymentMethod":{
      "displayName":"Visa 1234",
      "network":"Visa",
      "type":"1"
    },
    "transactionIdentifier":"xxxxxxxxxxxx",
    "paymentData":{
      "header":{
        "publicKeyHash":"xxxxxxxxxxxx",
        "ephemeralPublicKey":"xxxxxxxxxxxxxxxxxxxxx",
```

```
            "transactionId":"xxxxxxxxxxxxxx"
        },
        "signature":"xxxxxxx",
        "version":"EC_v1",
        "data":"xxxxxxxxxxx"
      }
    }
  }
```


4. Submit the payment request to PayDollar Server Side Direct Connection with the following parameter:
   **eWalletPaymentData**: Encoded Base64 PKPayment
   **eWalletService**: T
   **eWalletBrand**: APPLEPAY
5. PayDollar will send the data feed to merchant server for the final transaction result.

Related SDK document:
https://developer.apple.com/documentation/passkit/
https://developer.apple.com/library/archive/ApplePay_Guide/index.html#//apple_ref/doc/uid/TP40014764-CH1-SW1

## FPS

1. Merchant APP request FPS transaction and send request to Merchant Server
2. Merchant Server send the related order information to PayDollar Server Side Direct Connection.
3. PayDollar will return the FPS payment token at parameter "**errMsg**" with base64 format.
4. Merchant Generate an URL to allow the payment Application to retrieve the token. Merchant Server can decode base64 and store the FPS payment token or use API to retrieve the token from PayDollar.
    a. URL must be in HTTPS and with sub-domain "fps.", for example: https://fps.xxxxxx.xxx
    b. Sub-domain must be applied HKPost e-Cert and main domain name must be matched
    c. Merchant name at HKPost e-Cert must be **TOTALLY** matched with the Merchant FPS account name
    d. HKPost e-Cert must not be expired

    **Example**:

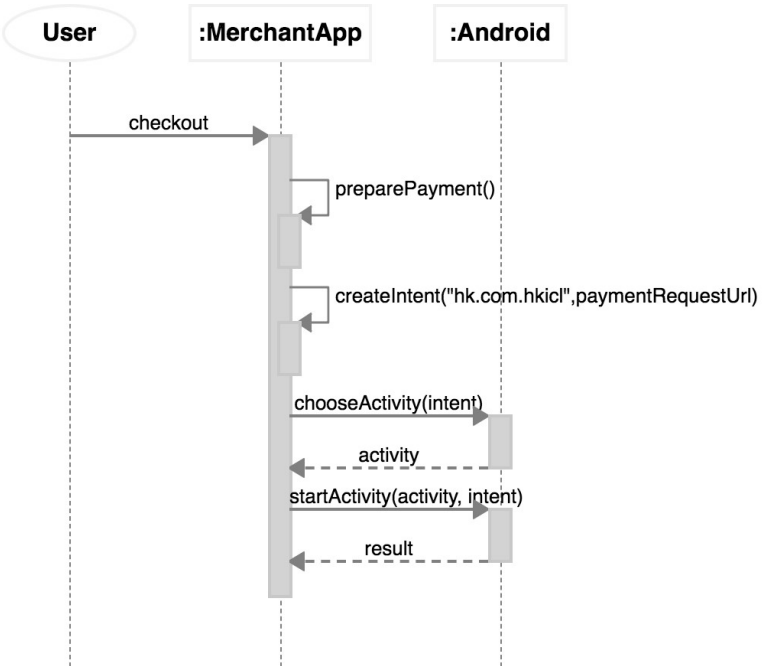
    *URL*: https://fps.merchant.com/order/123456

    *Response*:

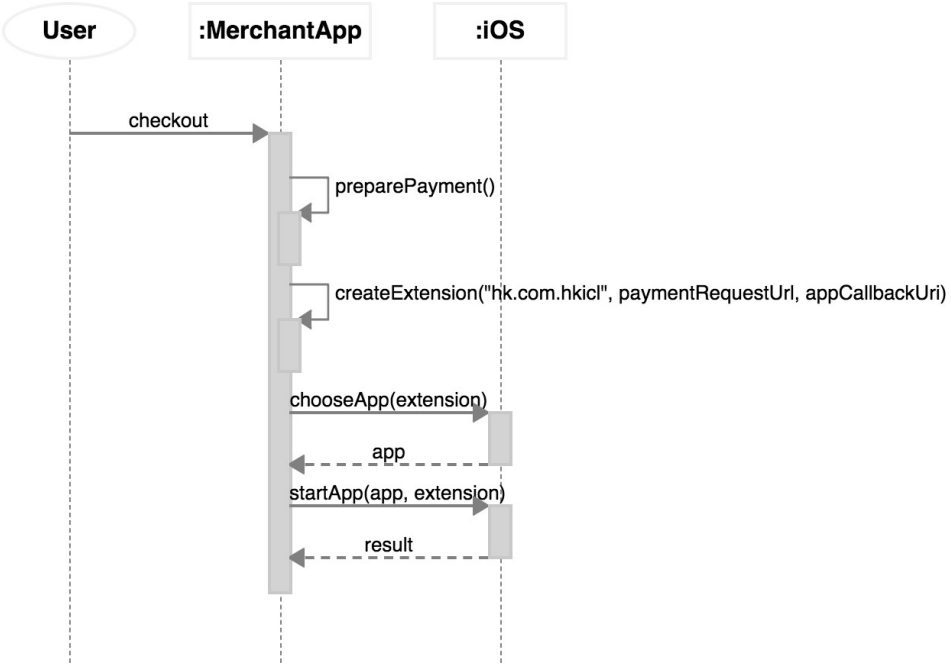
    00020101021230470115649032854431941021011103183130310111031831352047011530376454032.05802TH5908SCBTest6007BANGKOK62770110111031831303155181764896976980523020071403224139400000000713OON111031831363048814
5. Merchant Server return the URL to the Merchant APP and call payment request to the "hk.com.hkicl" with the above URL.
6. Supported Bank / Payment APP will be triggered and Client complete the payment.
7. PayDollar will send the data feed to merchant server for the final transaction result.

Merchant App Flow:

Android:




iOS:



## Octopus

1. Send the related order information to PayDollar Server Side Direct Connection.
2. PayDollar will return related transaction URI with Base64 format at parameter "**errMsg**".

   successful "**errMsg**" format:

   Base64(octopus payment URI)

   Example:

   data = octopus://payment?token=ysadsxs

   errMsg = b2N0b3B1czovL3BheW1lbnQ/dG9rZW49eXNhZHN4cw==
3. Decode the errMsg with Base64 to retrieve the Octopus payment URI.
4. Pass the Octopus payment URI to the application to trigger the payment.
5. PayDollar will send the data feed to merchant server for the final transaction result.

Sample Code:

Android:

```
public final static int OCTOPUS_APP_REQUEST_CODE = 10000; // This CODE is subject to
change by SI
List<PackageInfo> packs = getPackageManager().getInstalledPackages(0);
boolean hasOctopusApp = false;

for (int i = 0; i < packs.size(); i++) {
    PackageInfo p = packs.get(i);
    if (p.packageName.contains("com.octopuscards.nfc_reader")) {
        hasOctopusApp = true;
    }
}

if (hasOctopusApp) {
    // Make payment request to OOS
    // ……
    // method to get the URI in response data.
    String octopusuri = getOctopusURI(…..)
    // if installed package is OctopusApp
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri .parse(octopusuri));
    startActivityForResult(intent, OCTOPUS_APP_REQUEST_CODE);
} else {
    // don't have any octopus app. prompt to Google Play for download
    // ……
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == OCTOPUS_APP_REQUEST_CODE && resultCode == Activity.RESULT_OK)
{
        // Octopus returns here.
        // Merchant APP should get latest payment status from Payment gateway
    }
}
```

iOS:

```
// decode octopus payment token from the PayDollar response
let decodedData = Data(base64Encoded: str, options: .ignoreUnknownCharacters)
// sample decoded value : octopus://payment?token=1234567
let octopusPaymentUrl = String(data: decodedData!, encoding: .utf8)!

// build the return deeplink
let appName = Bundle.main.object(forInfoDictionaryKey: "CFBundleName") as! String
let appReturnUrl = appName + "://octopus/return"

// build the octopus payment url
let cURL = URL(string: octopusPaymentUrl! + "&return=" + appReturnUrl)!

// check octopus app installed or not
var canOpen : Bool = UIApplication.shared.canOpenURL(cURL)
if canOpen == true {
    UIApplication.shared.open(cURL, options: [:]) { (res) in
        print("Octopus app open response: \(res)")
    }
} else {
    // display app not installed or redirect client to App Store
}
```