Tap & Go Payment Integration Specification

Version: 1.0.46

Change Number	Revision Description	Version Number	Date	Remark
1.	Initial release.	1.0	8 April 2016	New release
2.				
3.				
4.				
5.				

Table of Contents

1 Introduction

This document is to illustrate the integration of Tap&Go payment interfaces to registered merchants.

There are several types of interfaces,

1. SDK for mobile apps (Android & iPhone)
2. Web browser based integration & API
3. File based interface for specific type of operation.

To ease mobile apps developer, SDK is provided to encapsulate the technical details of payment operation. Developers only need to include the library into their project file, follow the API specification to submit payment and retrieve payment result. However, Tap&Go app has to be installed in the end customer mobile to complete the whole payment operation.

For web based developers, integration guide and web-based API is provided and the integration flow is similar to industrial Online Payment Gateway. Again, Tap&Go app is also required to be installed in the customer's mobile handset. For customers using desktop browser, customers also need to have a handset with active Tap&Go app to complete the flow.

For file based interface, its primary purpose is to serve merchant with backend application servers to initialize payment request in batch mode via secure file transfer. Additional registration and information exchange with Tap&Go operation is required.

In summary, for all online payment, user is required to have a mobile device installed with Tap&Go app that pair with an activated virtual card (i.e. AIO SIM with embedded credit card) or plastic card. There is an authorized payment function in the app which allow user to authorize a payment request by input user's wallet PIN.

2 Prerequisite

To access Production and UAT platform, following conditions are required to be ready and approved by HKT Payment Limited or available before using this SDK:

1. Merchant Developer Account should be registered and approved. After approval, a unique Merchant ID was assigned to the developer account.
2. For each new application is created under the Merchant Developer Account, a unique App ID and corresponding App Secret key will be assigned.

3 Functional description

3.1 Merchant Authentication

- Each merchant is assigned with a unique merchant ID and public key to protect payment information. Also, each application (including mobile apps and web apps) is assigned with a unique App ID and App Secret Key.
- However, Merchant public key and App Secrete Key are sensitive information as they are essential to encrypt order information and authenticate the app itself respectively. Therefore, it is highly recommended to protect these keys from reverse engineering, e.g. code obscurations and split them into multiple data structure.
- Tap & Go will use the Merchant ID to retrieve Merchant Display Name during payment authorization screen.

3.2 Single Payment Authorization

3.2.1 Mobile App Integration

- It is required that Tap & Go app is installed and activated in the same device with the Merchant app so that Merchant app can send transaction data to Tap & Go app and trigger user's consent screen.
- Tap & Go App present a user's consent screen and then prompt for Wallet PIN. If the Wallet PIN is correct, Tap & Go app returns a token back to the SDK and SDK submits a payment request to its backend server.
- After the payment operation is done, the SDK authenticates the result by the App Key before returning result to Merchant app.
- Tap & Go Payment Server also provides Query Transaction API call so that merchant app or merchant server can validate the payment result is consistence with the result stored in Tap & Go Payment Server or in case the call to SDK is time out.

3.3 Recurrent Payment Authorization

- The authorization flow is similar to Single Payment Authorization.
- It also allows Merchant to get a customer's authorized payment token such that they can send recurrent payment requests.
- Upon successful authorization, Tap & Go returns a token to Merchant mobile app.
- Merchant is required to securely store this token without unnecessary disclosure.

3.4 Top-Up Request

- Merchant can also use the recurrent payment token to send Top-Up request to Tap & Go Payment Server.
- Tap & Go Server accepts server to server calls (i.e. with registered IP) only.

3.5 Token Administration

- Merchant is required to provide user interface to remove the recurrent payment in case customer want to revoke it.
- There is a Remove Token API in Tap & Go Payment Server for Merchant to complete this operation.

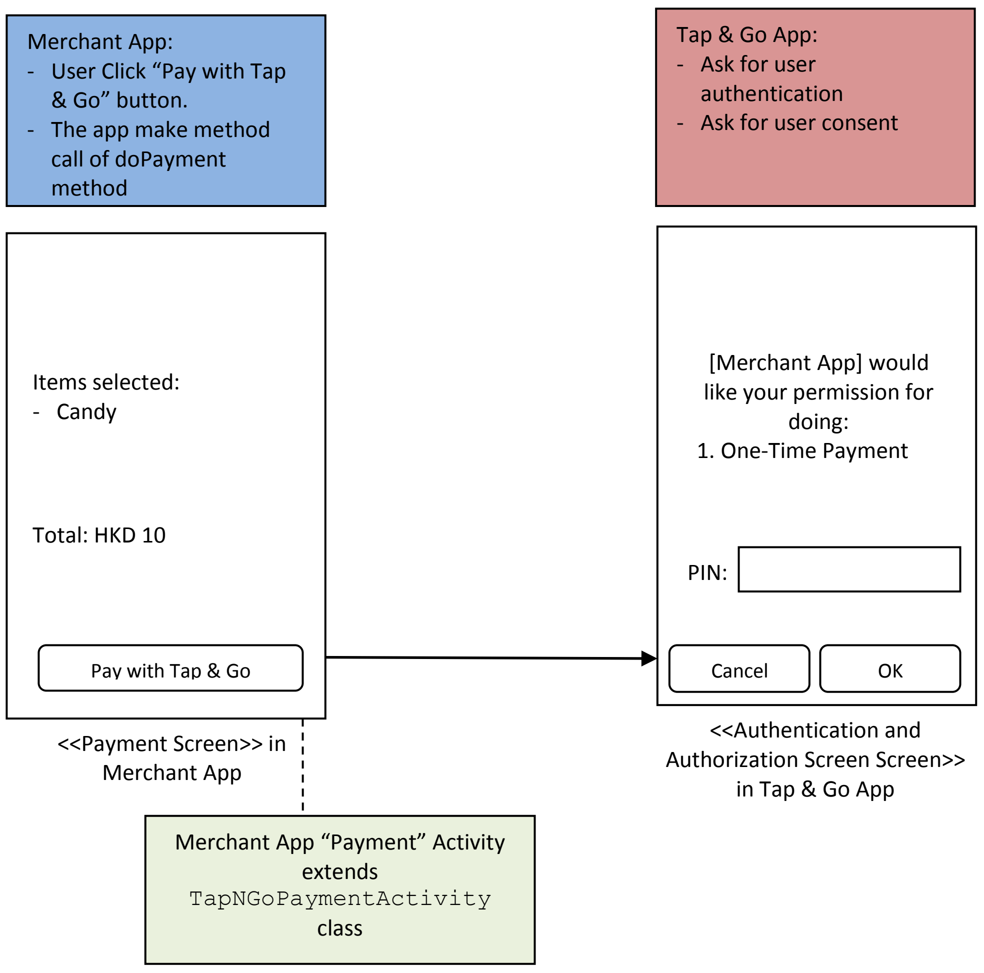
3.6 Transaction Query

- Merchant can re-confirm the status of payment result. Tap & Go Payment Server provides a unique transaction response ID to Merchant applications, Merchant is highly recommended to provide a unique transaction ID when they submit payment authorization request.

4 Interaction Flow

4.1 Flow for One-Time Payment

1. User Click “Pay with Tap & Go” button, which calls the SDK’s “doPayment” method.
2. In the “doPayment” method, it asks for user authentication and user consent on the payment to be processed by opening the Tap & Go app’s user consent screen.


Merchant App:
- User Click “Pay with Tap & Go” button.
- The app make method call of doPayment method
Items selected:
- Candy
Total: HKD 10
Pay with Tap & Go
<<Payment Screen>> in Merchant App
Merchant App “Payment” Activity extends TapNGoPaymentActivity class
Tap & Go App:
- Ask for user authentication
- Ask for user consent
[Merchant App] would like your permission for doing:
1. One-Time Payment
PIN:
Cancel
OK
<<Authentication and Authorization Screen Screen>> in Tap & Go App

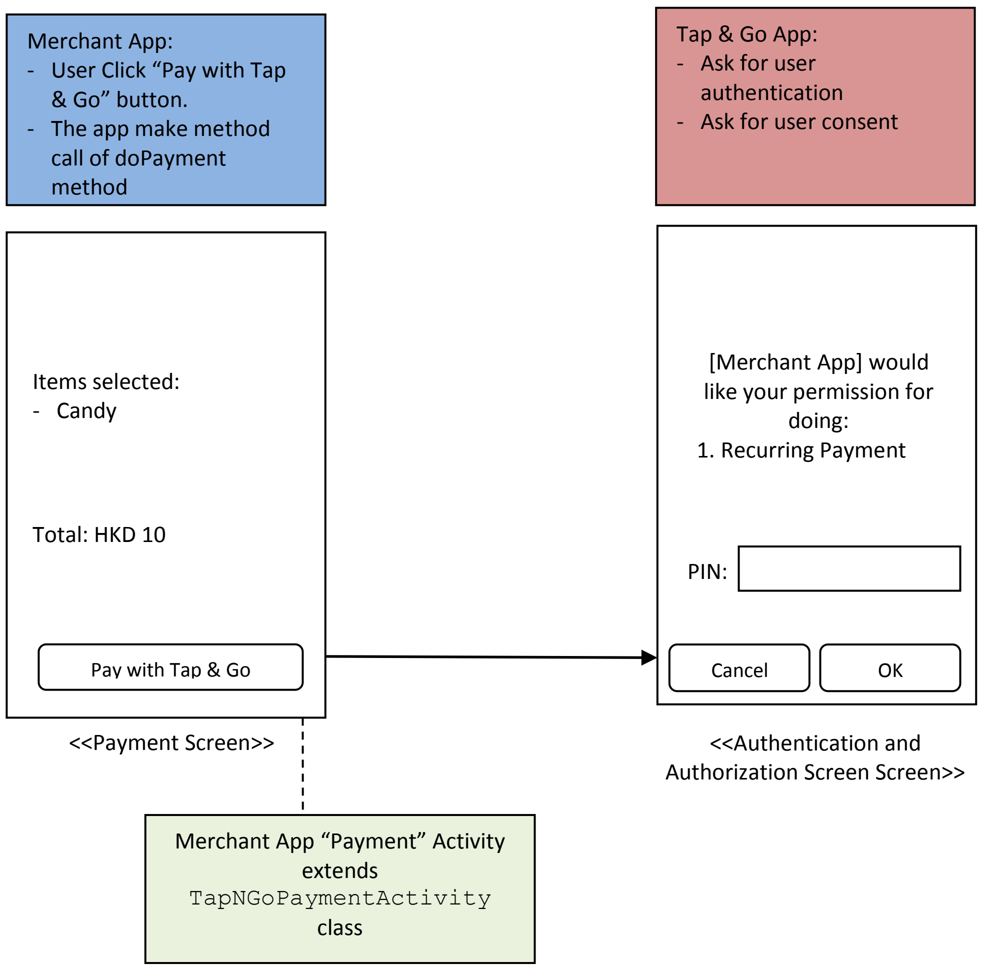
3. After user consent was confirmed, payment will be triggered.

4. SDK callbacks the Merchant App about the payment result in one of SDK's callback event below.
   1. On Payment Success
   2. On Payment Fail
   3. On Payment Error

   Merchant developer requires providing implementation (application logic for handling payment result) the above three callback method.

4.2 Flow for Recurring Payment

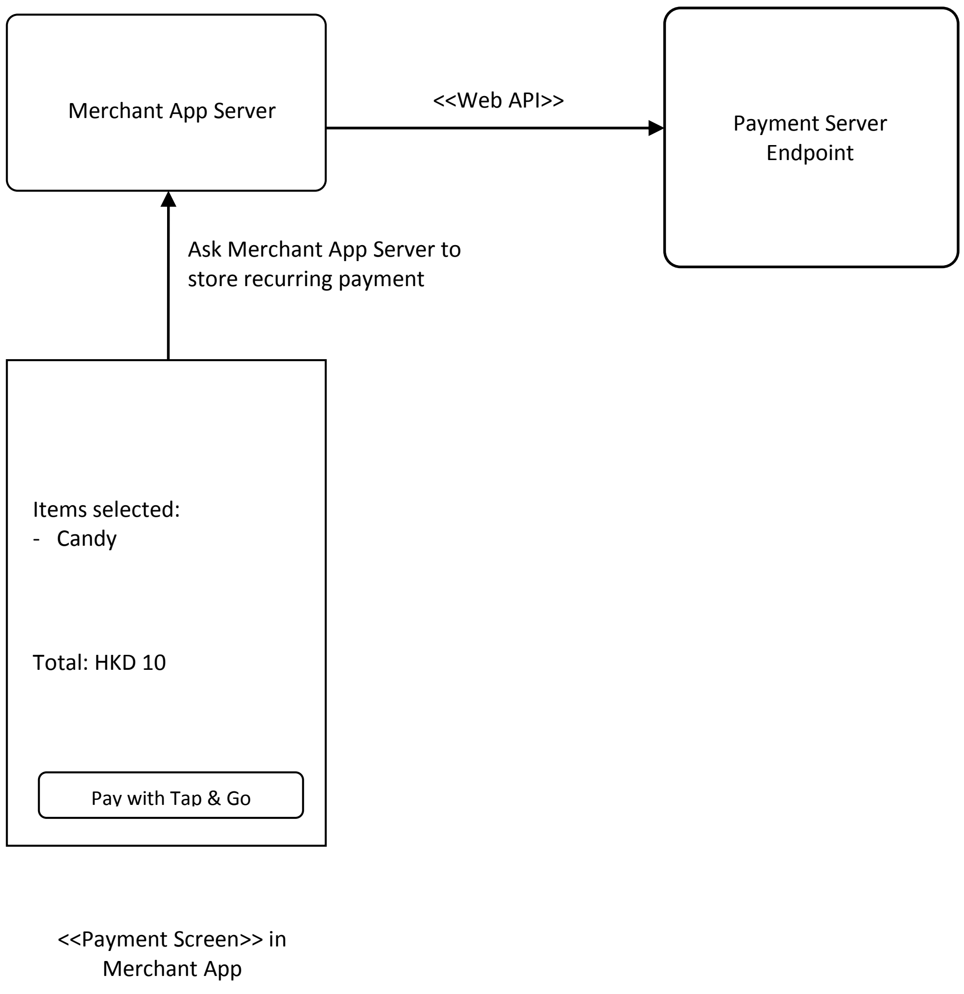
1. User Click "Pay with Tap & Go" button, which calls the SDK's "doPayment" method.
2. In the "doPayment" method, it asks for user authentication and user consent on the payment to be processed by opening the Tap & Go app's user consent screen.


Merchant App:
- User Click "Pay with Tap & Go" button.
- The app make method call of doPayment method
Tap & Go App:
- Ask for user authentication
- Ask for user consent
Items selected:
- Candy
Total: HKD 10
Pay with Tap & Go
<<Payment Screen>>
[Merchant App] would like your permission for doing:
1. Recurring Payment
PIN:
Cancel
OK
<<Authentication and Authorization Screen Screen>>
Merchant App "Payment" Activity extends TapNGoPaymentActivity class

3. After processing is complete, SDK callbacks the Merchant App about the payment result in one of SDK's callback event below.
   - On Payment Success
   - On Payment Fail
   - On Payment Error

   Merchant developer requires providing implementation (application logic for handling payment result) the above three callback method.

   In recurring payment, recurring token is returned to merchant app for it to ask merchant app server to store recurring token in secure way for future recurring payment triggering.



5 Android Integration

5.1 Platform Requirement

Require Android API level: 17 or higher

5.2 Integration Steps

Using Android studio (Require 2.0 or higher)

Option 1

1. Copy tapngosdk.aar to libs path
2. Add the following lines to bottom of build.gradle of app module (not project). (Notice: Tap&Go SDK is using oKHttp library, please also add this dependency, you may refer to the website http://square.github.io/okhttp/)

```
repositories {
   flatDir {
       dirs 'libs'
   }
}

dependencies {
    compile(name:'tapngosdk', ext:'aar')
    compile 'com.squareup.okhttp3:okhttp:3.3.1'
}
```

Option 2

1. Copy tapngosdk.aar to libs path
2. Click ‘File’ -> ‘New Module…’
3. Select ‘Import .JAR/.AAR Package’
4. Find and select tapngosdk.aar
5. Click ‘Finish'
6. Download okhttp library from http://square.github.io/okhttp/
7. Please the okhttp jar file to your project’s libs path
8. Wait for project sync

Using Eclipse (Require 4.0.0 or higher)

1. Unzip tapngosdk.aar
2. Rename the classes.jar to tangosdk.jar
3. Copy the tapngosdk.jar to your project’s libs path
4. Download okhttp library from http://square.github.io/okhttp/
5. Please the okhttp jar file to your project’s libs path
6. Clean project

5.3 Application Programming Interface (API)

Abstract Class: TapNGoPaymentActivity

Method	protected void doPayment (TapNGoPayment payment)		
Description	Set Recurrent Payment attributes		
Parameters	TapNGoPayment	payment	Payment object
Return	TapNGoPayResult	result	Object to store payment results

Method	protected abstract void onPaymentSuccess(TapNGoPayResult result)		
Description	Callback method to be overrided by Merchant App developer after making payment with successfully.		
Parameters	TapNGoPayResult	result	Object to store payment results

Method	protected abstract void onPaymentFail(TapNGoPayResult result)		
Description	Callback method to be overrided by Merchant App developer after fail to make payment.		
Parameters	TapNGoPayResult	result	Object to store payment results

Class: TapNGoPayment

Constructor

Method	public TapNGoPayment(String appId, String apiKey, String publicKey)		
Description	Get payment instance		
	String	appId	Application ID assign to each partner app
	String	apiKey	Unique secret key for each partner's application
	String	publicKey	Unique public key for each partner's application
Return	TapNGoPayment	TapNGoPayment instance	

Public Methods

Method	public void setSinglePayment(String merTradeNo, double totalPrice, String currency, String remark, String notifyUrl)		
Description	Set Single Payment attributes		
Parameters	String	merTradeNo	Unique ID for this transaction Format: alphanumeric Max. length: 64

	double	totalPrice	Price for the item
	String	currency	Must be HKD
	String	remark	Remark description to be shown, null value if no remark
	String	notifyUrl	URL provided by merchant to receive payment result notification, null value if no notifyUrl

Method	public void setRecurrentPayment(String merTradeNo, String currency, String remark)		
Description	Set Recurrent Payment attributes		
Parameters	String	merTradeNo	Unique ID for this transaction Format: alphanumeric Max. length: 64
	String	currency	Must be HKD
	String	remark	Remark description to be shown, null value if no remark

Method	public void setSingleAndRecurrentPayment(String merTradeNo, double totalPrice, String currency, String remark, String notifyUrl)		
Description	Set Single Payment attributes		
Parameters	String	merTradeNo	Unique ID for this transaction Format: alphanumeric Max. length: 64
	double	totalPrice	Price for the item
	String	currency	Must be HKD
	String	remark	Remark description to be shown, null value if no remark
	String	notifyUrl	URL provided by merchant to receive payment result notification, null value if no notifyUrl

Class: TapNGoPayResult

Constructor (default Constructor)
Public Methods

Method	public String getResultCode ()		
Description	Get result code of payment		
Parameters	NIL		
Return	String	result	Result code of payment

Method	public String getRecurrentToken ()		
Description	Get recurrent payment token		
Parameters	NIL		
Return	String	result	Recurrent payment token Null = No Token

Method	public String getMerTradeNo ()		
Description	Get Unique ID for this transaction		
Parameters	NIL		
Return	String	result	Merchant trade number Null = No merchant trade number

Method	public String getTradeStatus ()		
Description	Get Unique ID for this transaction		
Parameters	NIL		
Return	TradeStatus	result	Trade Status **TRADE_FINISHED** – payment success **TRADE_CLOSED** – payment cancelled / failed **WAIT_TO_PAY** – payment is processing **UNKNOWN** – payment is unknown

Method	public String getMessageEn ()		
Description	Get returned message in English		
Parameters	NIL		
Return	String	result	Returned message in English (For reference only) Null = No message

Method	public String getMessageZh ()		
Description	Get returned message in Chinese		
Parameters	NIL		
Return	String	result	Returned message in Chinese (For reference only) Null = No message

Class: TapNGoSdkSettings

Public Methods

Method	public static String getSdkVersion ()
Description	Get SDK version number

Parameters	NIL		
Return	String	result	Returned SDK version number in x.y.z format

Method	public static boolean isSandboxModeEnabled ()		
Description	Check is sandbox mode enabled		
Parameters	NIL		

Return	boolean	result	true = Sandbox mode enabled
			false = Sandbox mode disabled

Method	public static void setSandboxMode(boolean enable)		
Description	Set sandbox mode (default = false)		

Parameters	boolean	enable	true = Enable sandbox mode
			false = Disable sandbox mode

Return	NIL		

5.4 Sample codes

1. Inherit TapNGoPaymentActivity

```
public class MyActivity extends TapNGoPaymentActivity { … }
```

2. Override callback methods

```
  @Override
  protected void onPaymentSuccess(TapNGoPayResult payResult) {
  // Handle payment success here
  }

  @Override
  protected void onPaymentFail(TapNGoPayResult payResult) {
  // Handle payment fail here
  }
```

3. Configure SDK settings

```
TapNGoSdkSettings.setSandboxMode(true);
```

4. Call payment methods

```
private static final String APP_ID = "dummy_app_id";
private static final String API_KEY = "dummy_api_key";
private static final String PUBLIC_KEY = "dummy_public_key";
private static final String MER_TRADE_NO = "dummy_mertradeno";

private static final double totalPrice = 100;
private static final String currency = "HKD";
private static final String remark = "remark";
private static final String notifyUrl = "notifyUrl"



private void doSinglePayment() {
    TapNGoPayment payment = new TapNGoPayment(APP_ID, API_KEY, PUBLIC_KEY);
    payment.setSinglePayment(MER_TRADE_NO, totalPrice, currency, remark, 
notifyUrl);
    doPayment(payment);
}

private void doRecurrentPayment() {
    TapNGoPayment payment = new TapNGoPayment(APP_ID, API_KEY, PUBLIC_KEY);
    payment.setRecurrentPayment(MER_TRADE_NO, currency, remark);
    doPayment(payment);
}

private void doSingleRecurrentPayment() {
    TapNGoPayment payment = new TapNGoPayment(APP_ID, API_KEY, PUBLIC_KEY);
    payment.setSingleAndRecurrentPayment(MER_TRADE_NO, totalPrice, currency, 
remark, notifyUrl);
    doPayment(payment);
}
```

5.5 Error codes

Please refer to SDK error codes in appendix 10.1.

6 iOS Integration

6.1 Platform Requirement

Require iOS version: 7.0 or higher
Libraries:
(Starting from SDK 1.2.0)
i. AFNetworking 3.x.x
ii. CocoaLumberjack 2.x.x or 3.x.x
(SDK 1.1.0 or previous version)
iii. AFNetworking 2.x.x
iv. CocoaLumberjack 2.x.x

6.2 Integration Steps

1. Install CocoaPods by entering sudo gem install cocoapods on terminal (Refer to https://cocoapods.org/)
2. Go to working copy then enter pod init on terminal
3. Open generated Podfile, edit it by entering the following line

```
target '{your_project_target}' do
  pod 'AFNetworking', '~> 3.0'
  pod 'AFXMLDictionarySerializer'
  pod 'CocoaLumberjack'
 end
```

4. Go back to terminal, enter pod install. Then {your_project_name}.xcworkspace will be generated. You should do your development on this workspace.
5. Double click to open {your_project_name}.xcworkspace.
6. Copy tapngosdk.framework to your project folder.
7. In the project navigator on Xcode, select the project or group within a project to which you want to add the framework.
8. Choose File > Add Files to "<App_Name>".
9. Select the tapngosdk.framework, and click Add.
10. Select Project on project navigator.
11. Select Target of your project.
12. Add the path of tapngosdk.framework in Framework Search Paths.
13. Add following code into App's info plist

```
<key>CFBundleURLTypes</key>
        <array>
                <dict>
                        <key>CFBundleTypeRole</key>
                        <string>Editor</string>
                        <key>CFBundleURLName</key>
                        <string>your_bundle_id</string>
                        <key>CFBundleURLSchemes</key>
                        <array>
                                <string>tapngo{your_call_back_id}</string>
```

```
                    </array>
             </dict>
      </array>

<key>LSApplicationQueriesSchemes</key>
      <array>
             <string>tapngowallet</string>
      </array>
```

14. Add following code into App's delegate

```
- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url {

    [[TGSDKAppDelegate sharedInstance] application:application handleOpenURL:url];
    return YES;
}

- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url
options:(NSDictionary<NSString*, id> *)options {

    [[TGSDKAppDelegate sharedInstance] application:app handleOpenURL:url];
    return YES;
}
```

6.3 Application Programming Interface (API)

Protocol: TGSDKPaymentResultDelegate

Required protocol

Method	- (void)doPaymentSuccessWithPayResult:(TGSDKPayResult*)payResult		
Description	Call back protocol for success case		
Parameters	TGSDKPayResult*	payResult	Object to store payment results

Method	- (void)doPaymentFailWithPayResult:(TGSDKPayResult*)payResult		
Description	Call back protocol for success case		
Parameters	TGSDKPayResult*	payResult	Object to store payment results

Method	- (void)doPaymentErrorWithPayResult:(TGSDKPayResult*)payResult		
Description	Call back protocol for success case		
Parameters	TGSDKPayResult*	payResult	Object to store payment results

Class: TGSDKPaymentManager

Constructor

Method	- (instancetype) initWithDelegate:(id<TGSDKPaymentResultDelegate>)delegate		
Description	Get TGSDKPayment instance		
Parameters	Id<TGSDKPaymentResultDelegate>	delegate	Class which implemented TGSDKPaymentResultDelegate
Return	TGSDKPaymentManager*	TGSDKPaymentManager instance	

Public Methods

Method	- (void) doPayment:(TGSDKPayment*)payment		
Description	Set Recurrent Payment attributes		
Parameters	TGSDKPayment*	payment	Instance of TGSDKPayment

Class: TGSDKPayment

Constructor

Method	- (instancetype) initWithAppId:(NSString *)appId apiKey:(NSString *) apiKey publicKey:(NSString*)publicKey callBackId:(NSString *)callBackId		
Description	Get payment instance		
	NSString*	appId	Application ID assign to each partner app
	NSString*	apiKey	Unique secret key for each partner's application
	NSString*	publicKey	Unique public key for each partner's application
	NSString*	callBackId	Unique call back id for each partner's application.
Return	TGSDKPayment*	TGSDKPayment instance	

Public Methods

Method	- (void) setSinglePaymentWithMerTrade:(NSString *)merTradeNo totalPrice:(double)totalPrice currency:(NSString *)currency remark:(NString *)remark notifyUrl:(NSString *)notifyUrl		
Description	Set Single Payment attributes		
Parameters	NSString*	merTradeNo	Unique ID for this transaction Format: alphanumeric Max. length: 64
	double	totalPrice	Price for the item
	NSString*	currency	Must be HKD
	NSString*	remark	Remark description to be shown, null value if no remark

	NSString*	notifyURL	URL provided by merchant to receive payment result notification, null value if no notifyUrl

Method	- (void) setRecurrentPaymentWithMerTradeNo:(NSString *)merTradeNo currency(NSString *)currency remark:(NString *)remark		
Description	Set Recurrent Payment attributes		
Parameters	NSString*	merTradeNo	Unique ID for this transaction Format: alphanumeric Max. length: 64
	NSString*	currency	Must be HKD
	NSString*	remark	Remark description to be shown, null value if no remark

Method	- (NSInteger) setSingleAndRecurrentPaymentWithMerTrade:(NSString *)merTradeNo totalPrice:(double)totalPrice currency:(NSString *)currency remark:(NString *)remark notifyUrl:(NSString *)notifyUrl		
Description	Set Single Payment attributes		
Parameters	NSString*	merTradeNo	Unique ID for this transaction Format: alphanumeric Max. length: 64
	double	totalPrice	Price for the item
	NSString*	currency	Must be HKD
	NSString*	remark	Remark description to be shown, null value if no remark
	NSString*	notifyUrl	URL provided by merchant to receive payment result notification, null value if no notifyUrl

Class: TGSDKPayResult

Constructor (default Constructor)
Public Methods

Method	-(NSString *) getResultCode		
Description	Get result code of payment		
Parameters	NIL	Parameters	NIL
Return	NSString *	Return	String

Method	-(NSString *) getRecurrentToken
Description	Get recurrent payment token

Parameters	NIL		
Return	NSString *	result	Recurrent payment token Null = No Token

Method	-(NSString *) getMerTradeNo		
Description	Get Unique ID for this transaction		
Parameters	NIL		
Return	NSString *	result	Merchant trade number Null = No merchant trade number

Method	-(TGSDKPayResultTradeStatus) getTradeStatus		
Description	Get Unique ID for this transaction		
Parameters	NIL		
Return	TGSDKPayResultTradeStatus	result	Trade Status **TGSDKPayResultTradeStatusTradeFinished** – payment success **TGSDKPayResultTradeStatusTradeClosed** – payment cancelled / failed **TGSDKPayResultTradeStatusTradeWaitToPlay** – payment is processing **TGSDKPayResultTradeStatusTradeUnknown** – payment is unknown

Method	- (NSString *) getMessageEn		
Description	Get returned message in English		
Parameters	NIL		
Return	NSString *	result	Returned message in English (For reference only) Null = No message

Method	- (NSString *) getMessageZh		
Description	Get returned message in Chinese		
Parameters	NIL		
Return	NSString *	result	Returned message in Chinese (For reference only) Null = No message

Class: TGSDKAppDelegate

Constructor (default Constructor)
Public Methods

Method	- (BOOL) application:(UIApplication *)application handleOpenURL:(NSURL *)url

Description	Asks the delegate to open a resource identified by URL. Call it at your app delegate's method with same name		
Parameters	UIApplication *	application	Your singleton app object.
	NSURL *	url	A object representing a URL (Universal Resource Locator). See the appendix of App Programming Guide for iOS for Apple-registered schemes for URLs.
Return	BOOL	result	YES = if the delegate successfully handled the request NO = if the attempt to handle the URL failed.

Method	- (BOOL)application:(UIApplication *)*app* openURL:(NSURL *)*url* options:(NSDictionary<NSString *, id> *)*options*		
Description	Asks the delegate to open a resource specified by a URL, and provides a dictionary of launch options. Call it at your app delegate's method with same name		
Parameters	UIApplication *	application	Your singleton app object.
	NSURL *	url	The URL resource to open. This resource can be a network resource or a file. For information about the Apple-registered URL schemes.
	NSDictionary<NSString *, id> *	options	A dictionary of launch options.
Return	BOOL	result	YES = if the delegate successfully handled the request NO = if the attempt to handle the URL failed.

Class: TGSDKSettings

Public Methods

Method	+ (NSString*)getSdkVersion		
Description	Get SDK version number		
Parameters	NIL		
Return	NSString*	result	Returned SDK version number in x.y.z format

Method	+ (BOOL)isSandBoxModeEnabled
Description	Check is sandbox mode enabled

Parameters	NIL		

Return	BOOL	result	true = Sandbox mode enabled
			false = Sandbox mode disabled

Method	+ (void)setSandBoxModeEnable:(BOOL)enable		
Description	Set sandbox mode (default = false)		

Parameters	BOOL	enable	true = Enable sandbox mode
			false = Disable sandbox mode

Return	NIL		

6.4 Sample Codes

1. Import SDK manager header

```
#import <tapngosdk/TGSDKPaymentManager.h>
```

2. Implement delegate

```
// In .h file

@interface ViewController : UIViewController <TGSDKPaymentResultDelegate>

// In .m file

- (void)doPaymentSuccessWithPayResult:(TGSDKPayResult *)payResult {

}

- (void)doPaymentFailWithPayResult:(TGSDKPayResult *)payResult {

}
```

3. Configure SDK settings

```
[TGSDKSettings setSandBoxModeEnable:YES];
```

4. Call single payment

```
    TGSDKPayment *payment = [[TGSDKPayment alloc] initWithAppId:@"your_app_id"
apiKey:@"your_api_key" publicKey:@”your_public_key”
callBackId:@”your_callback_id”];

    [payment setSinglePaymentWithMerTradeNo:@"MerTradeNo" totalPrice:100.00
currency:@"HKD" remark:@”remark” notifyUrl:@”notifyUrl”];

    self.paymentManager = [[TGSDKPaymentManager alloc] initWithDelegate:self];
    [self.paymentManager doPayment:payment];
```

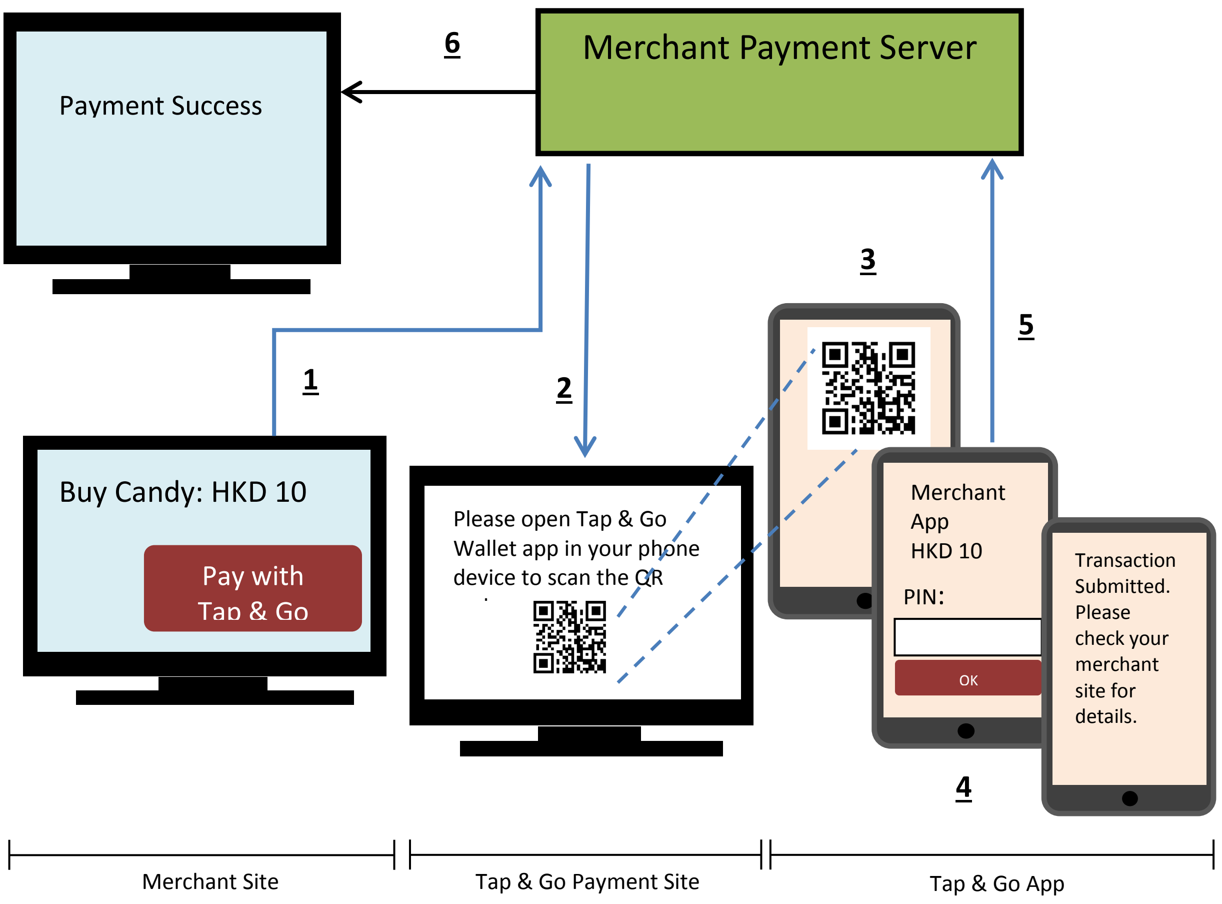
6.5 Error codes

Please refer to SDK error codes in appendix 10.1.

7 Web Payment Integration

7.1 Interaction Flow

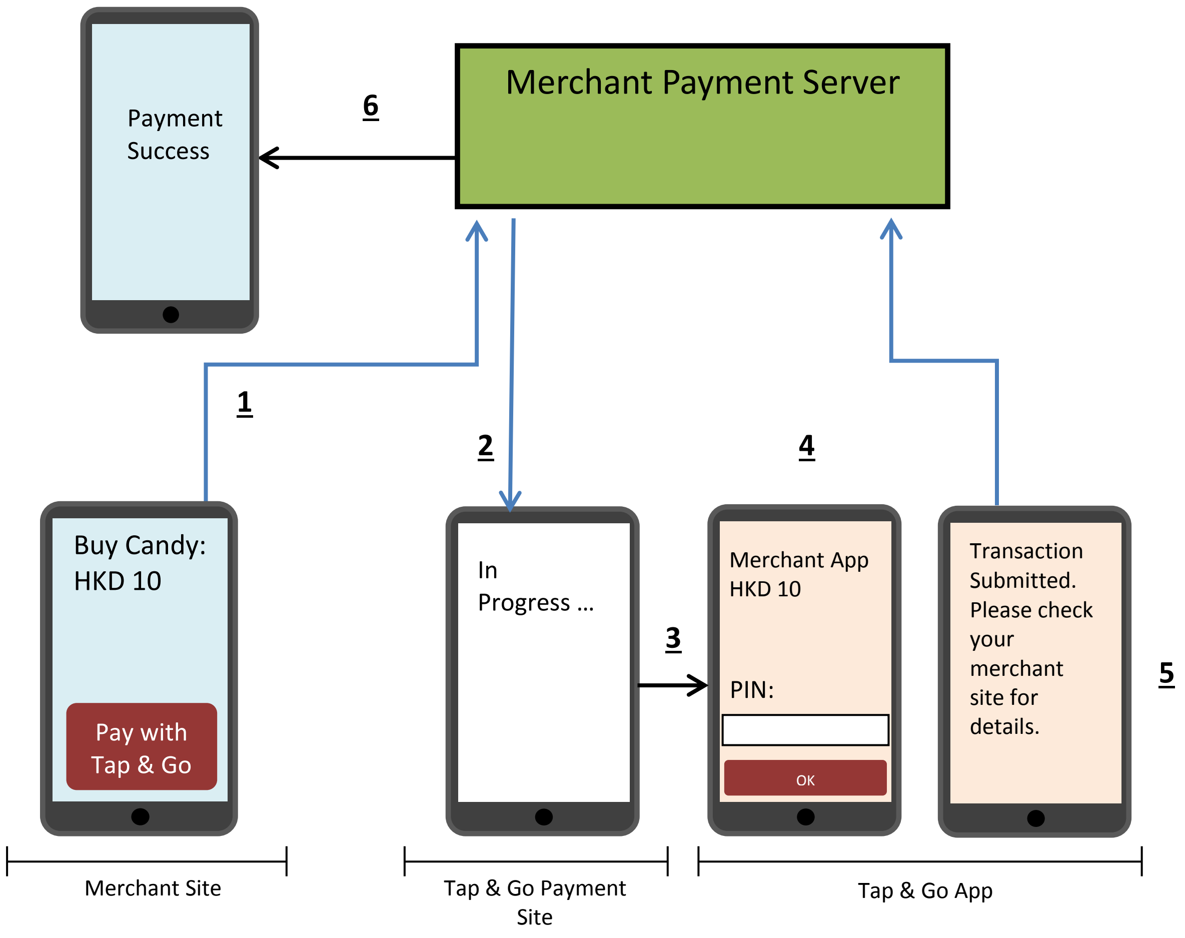
7.1.1 Flow when user using desktop to visit the merchant site



Steps:

1. User goes to the merchant website on desktop browser and click 'Pay with Tap & Go' button.
2. A page will be displayed with a QR code.
3. User opens Tap & Go mobile app and scan the QR code.
4. Transaction information will be displayed on Tap & Go mobile app.
5. User enters PIN to authorize the payment.
6. Upon payment finished, desktop browser redirects to a result page that is hosted by merchant.

7.1.2 Flow when user using mobile to visit the merchant site



Steps:

1. User goes to the merchant website on mobile browser and click 'Pay with Tap & Go' button.
2. A page will be displayed that launches Tap & Go app.
3. Transaction information will be displayed on Tap & Go mobile app.
4. User enters PIN to authorize the payment.
5. A screen will be displayed on Tap & Go mobile app, instructing the user to switch back to browser app.
6. Upon payment finished, browser redirects to a result page that is hosted by merchant.

7.2 Application Programming Interface (API)

7.2.1 Basic Information and Settings for Production

Field	**Variable Name in URL Path**	**Value**
Server Domain:	{SERVER_DOMAIN}	gateway.tapngo.com.hk

7.2.2 Basic Information and Settings for UAT

Field	**Variable Name in URL Path**	**Value**
Server Domain:	{SERVER_DOMAIN}	gateway.sandbox.tapngo.com.hk

7.2.3 Perform Payment Endpoint

To submit a payment, include following parameters in Merchant checkout form:

API Path	https://{SERVER_DOMAIN}/web/payments
Method	POST

Parameter	**Type**	**Description**	**Mandatory**
appId	String(10)	Application ID	Y
merTradeNo	String(64)	Unique ID for this transaction Format: alphanumeric Max. length: 64	Y
paymentType	String	Payment type of this transaction **S** – Single, one-off **R** – Recurrent **SR** – Both single and recurrent	Y
payload	String	Encrypted payment data in JSON format. Please refer to section 8.5 for content details (e.g. return URL, amount, etc.) and section 8.6 for encryption specification.	Y
extras	String	Optional. Encrypted extra data in JSON format. E.g. if user name is required too: {“name”:”Chan Tai Man”} Please refer to section 8.6 for encryption specification.	N
transactionType	String	CR – Purchase (by Tap & Go user to merchant) DB – Top-up (from merchant to Tap & Go user) DC – Both (**ONLY applicable when paymentType=R**) Default – CR	N

sign	String	HMAC-SHA512 of all fields above	Y

Sample code:

```
<form action="https://gateway.tapngo.com.hk/web/payments" method="post">
        <input type="hidden" name="appId" value="1234567890"/>
        <input type="hidden" name="merTradeNo"
value="72640892610489271356077" />
        <input type="hidden" name="paymentType" value="S"/>
        <input type="hidden" name="payload"
value="DJAWOIDJIODJWOCINJIOja8728ujslejicjC6JIFOji91CJN1jcpbfdoJIOhuIIOHoG
IOY8927f33f2g92"/>
        <input type="hidden" name="extras"
value="faejo61dfhFW0awd8j2Fo12odja8dHiOFwuwdjaFuiw27DnfiuUo2odah8wDFd8DAD9
012"/>
        <input type="hidden" name="sign"
value="DWAodjawiowjdiwoijwacnwi7c2a8whyur82qcu8q89qu8cu9CNNCDHFn28nc98cnu8
"/>
        <p><input type="submit" value="Pay By Tap &amp; Go"/></p>
</form>
```

7.2.4 Return URL - Payment Result Callback

Upon the transaction has finished, Tap & Go server will trigger a callback to merchant's website (i.e. browser redirect) to notify the payment result via form post. This URL should use POST and accept the following parameters in media type **"application/x-www-form-urlencoded"**:

Parameter	**Type**	**Description**	**Mandatory**
merTradeNo	String(64)	Unique ID for this transaction	Y
tradeNo	String	Tap & Go transaction ID	Y for paymentType = S / SR
tradeStatus	String	Status for this transaction. TRADE_FINISHED – payment success TRADE_CLOSED – payment cancelled / failed WAIT_TO_PAY – payment is processing	Y for paymentType = S / SR
recurrentToken	String	Token for recurrent payment	Y for paymentType = R / SR
msg	String	Message	Y
resultCode	String	Result code of the operation	Y
sign	String	SHA256 of all fields above	Y

Example code that triggers the callback for single payment (paymentType = S):

```
<form action="https://your.domain.com/callback_url" method="post">
  <input type="hidden" name="merTradeNo" value="283891071823583014"/>
  <input type="hidden" name="tradeNo" value="1958928931723852023957013582785"/>
  <input type="hidden" name="tradeStatus" value="S"/>
  <input type="hidden" name="msg" value="SUCCESS"/>
  <input type="hidden" name="resultCode" value="0"/>
  <input type="hidden" name="sign"
         value=" I29fjdWoVajfw9W720Fjaow9A02nV72isaoqo2p6xwED"/>
</form>
```

Example code that triggers the callback for obtaining recurrent token (paymentType = R):

```
<form action="https://your.domain.com/callback_url" method="post">
  <input type="hidden" name="merTradeNo" value="283891071823583014"/>
  <input type="hidden" name="recurrentToken"
         value="B8baapFW8Fwj9a01jWPFj2pFowsap8Fw7f89"/>
  <input type="hidden" name="msg" value="SUCCESS"/>
  <input type="hidden" name="resultCode" value="0"/>
  <input type="hidden" name="sign"
         value=" I29fjdWoVajfw9W720Fjaow9A02nV72isaoqo2p6xwED"/>
```

```
</form>
```

Example code that triggers the callback for both single payment and obtaining recurrent token (paymentType = SR):

```
<form action="https://your.domain.com/callback_url" method="post">
  <input type="hidden" name="merTradeNo" value="283891071823583014"/>
  <input type="hidden" name="tradeNo" value="1958928931723852023957013582785"/>
  <input type="hidden" name="tradeStatus" value="S"/>
  <input type="hidden" name="recurrentToken"
         value="B8baapFW8Fwj9a01jWPFj2pFowsap8Fw7f89"/>
  <input type="hidden" name="msg" value="SUCCESS"/>
  <input type="hidden" name="resultCode" value="0"/>
  <input type="hidden" name="sign"
         value=" I29fjdWoVajfw9W720Fjaow9A02nV72isaoqo2p6xwED"/>
</form>
```

Signature is calculated by joining all received parameters into query string in alphabetical order and perform SHA256 on it. For more details about the query string format, please refer to section 8.8.1.

8 Web API Integration

8.1 Requirement

The Merchant servers' fixed IP(s) are registered or updated.

8.2 Basic Information and Settings for Production

Field	Variable Name in URL Path	Value
Server Domain:	{SERVER_DOMAIN}	gateway2.tapngo.com.hk

8.3 Basic Information and Settings for UAT

Field	Variable Name in URL Path	Value
Server Domain:	{SERVER_DOMAIN}	gateway.sandbox.tapngo.com.hk

8.4 Content Negotiation

8.4.1.1 "Accept" header

"Accept**"** header in the request is used for content negotiation. Modify the value to change the response type accordingly.

Request's "Accept" Header Value	Response Type
application/xml	application/xml
application/json	application/json

8.4.1.2 "Content-Type" header

Web API accept the "Content-Type" header value as below:

Content-Type: application/x-www-form-urlencoded

8.5 Payment Information

Some requests are required to provide "**payload**" as a mandatory parameter. "**payload**" is the RSA encrypted string of the payment information "**paymentInfo**". "**paymentInfo**" is the payment information that constructed in JSON format. All fields are in String format. If paymentType is S or SR, "merTradeNo" is needed. The format is alphanumeric with maximum length of 64 characters. "**lang**" parameter accept "en" for English and "zh" for Chinese.

Different "**paymentType**" values will have different contents in "**paymentInfo**".

Single payment (paymentType = S / SR) or **Do Recurrent Payment API** (Section 8.9.1):
{"totalPrice":"500.00",
"currency":"HKD",
"merTradeNo":"1234567890123456789012345678901234567890123456
78901234",
"notifyUrl":"https://merchant.servers.domain.com/merchant/part/payment/some/pa
th/leads/notify",
"returnUrl":"https://merchant.servers.domain.com/merchant/part/payment/some/pa
th/leads/return",
"remark":"This user has special request",
"lang":"en"}

Recurrent payment (paymentType = **R**):
{“currency”:”HKD”
"returnUrl":"https://merchant.servers.domain.com/merchant/part/payment/some/pa
th/leads/return",
"remark":"This user has special request",
"lang":"en"
}

8.6 RSA Encryption

For the parameters that required to encrypt using RSA public key, the following information will be used.

Algorithm:
RSA/ECB/OAEPWithSHA-1AndMGF1Padding
Public Key:
The given RSA public key is Base64 Encoded, with 4096 bit key size

8.7 Optional Parameters

Do not pass empty value into optional parameters. These kinds of parameters should be neglected directly.

Example:

Request Parameter	**Value**
key1	value1
key2	
key3	value3

Post body:
Incorrect:

key1=value1&key2=&key3=value3

Correct:
key1=value1&key3=value3

8.8 Signature

For the request and response of all APIs, there is a “sign” field. It is used for checking the integrity and correctness of the transferred information.

Signature is generated by first hashing the parameter value with **HMAC-SHA512** and using “API Key” as the **hash secret**, and then encoding the result bytes to a **Base64 string**.

8.8.1 Request Signature

1. Arrange parameters in alphabetical order
2. **Exclude all parameters that are null**
3. Join all request parameters in form of query string *key1=value1&key2=value2….*
4. Hash the query string with **HMAC-SHA512**
   a. Remark: API Key and the query string to be hashed should be encoded in UTF-8
5. Signature is the hash result encoded with **Base64**

Example: Query payment status API

Request Parameter	**Value**
appId	1234567890
merTradeNo	9876543210
timestamp	140235441215746

Query string:
appId=1234567890&merTradeNo=9876543210&timestamp=140235441215746

API key:
025a0h7L65Ewx+27fRjN3ILHwlfw9Ccz5/Cv6bb5oKKU+MnRoKeT/cvgsDWV7O7ZNLJj W8tlqolyV3fH3sjxrQ==

Signature generated:
EYxFzibGM83t+5tXzu47siL7b2JKsFwHH4LRjaEmMkdwgo/12g8F11C61XDA5WZLsLeqP 399GRc2uCoGfHCmHg==

8.8.2 Response Signature

All responses are return in a common format. The content part (highlighted in **red** in the sample below) is used for calculating the signature. Steps to generate the response signature from the received response could be found as below.

Step 1: construct the string for sign generation from response
E.g. String to generate signature: {"chiMessage":"請求完成","engMessage":"Request Success","internal":"Request Success","payload":{"merTradeNo":"123","tradeNo":"12345678901234","tradeStatus":"TRADE_FINISHED"},"resultCode":"0"}

Step 2: generate hast by using HMAC-SHA512
sign = HMAC-SHA512(<api key>, <string from step 1>)

Response Sample:

```
<result>
        <content>
                <resultCode>0</resultCode>
                <chiMessage>請求完成</chiMessage>
                <engMessage>Request success</engMessage>
                <internal>Request success</internal>
                <payload>
                        …
                </payload>
        </content>
        <sign>150411BaJ123JHGGggB</sign>
</result>
```

```
{
  “content”: {
    “resultCode”: “0”,
    “chiMessage”: “請求完成”,
    “engMessage”: “Request success”,
    “internal”: “Request success”,
    “payload”: {
      …
    }
  },
  “sign”: “150411BaJ123JHGGggB”
}
```

8.9 API Specification

8.9.1 Do Recurrent Payment

Description:	Request to perform a recurrent payment with using the “recurrentToken” and other information
Request	POST

Method:	
URL:	https://{SERVER_DOMAIN}/paymentApi/payment/recurrent

Request Parameters:

Parameter	***Mandatory***	***Example***	***Remarks***
appId	Y	70136705	App ID of the Merchant App
recurrentToken	Y	kis8lk8yt4wetez5mkd8l…	Token for recurrent payment
payload	Y	2gWXhvpU8S7iPNTNnn…	RSA(paymentInfo) Encrypted payment data in JSON format. Please refer to section 8.5 for content details (e.g. return URL, amount, etc.) and section 8.6 for encryption specification.
transactionType	N	CR	CR – Purchase (by Tap & Go user to merchant) DB – Top-up (from merchant to Tap & Go user) Default – CR
timestamp	Y	140235441215746	Unix Timestamp
sign	Y	dk8ued893j…	Signature

Response Fields:

Field	***Example***	***Remarks***
resultCode	0	Result code
message	Request Success	Result message
merTradeNo	12915236	A numbered transaction ID (Given by merchant. Unique per the merchant) Format: alphanumeric Max. length: 64
tradeNo	150423JHGgB	Tap & Go Transaction ID
tradeStatus	TRADE_FINISHED	Current status of this payment request

		TRADE_FINISHED – payment success TRADE_CLOSED – payment cancelled / failed WAIT_TO_PAY – payment is processing
sign	sd9e4hf684…	Signature

Possible Result:

Result Code	***Reference Code***	***Description***
0		Request Success
400		Bad request • Mandatory parameters are not provided • Unable to decrypt payload • Method not allowed
403		Forbidden • Incoming IP not registered
461		Invalid timestamp • The provided timestamp is outside of the valid API time period
462		Invalid Signature • Signature is not matched with the request parameters
489		Invalid information • Cannot obtain apiKey to check signature
490		Payment failed (Ref: xxxx)
	ZA03	Illegal parameters
	ZA185	Illegal parameters
	ZA186	Currency invalid (Currently only “HKD” is supported)
	ZA187	MerTradeNo is repeated
	ZA188	Payment service is not activated for this partner ID
	ZA190	Merchant not allowed to credit customers’ Accounts
	ZA191	Credit TopUp Failure
	D01	Token not found
	D02	This token has been disabled / revoked
	D03	Token has expired
	F01	Profile not found (IMSI)
	F05	Account No Active Card Exception
	F07	Account No Active Mobile number Exception
	444	Service unavailable
	445	Service unavailable
	446	Service unavailable
	447	Service unavailable
	993	Decryption failed
	994	Decrypted data is in unknown format

	995	Merchant is not active
	996	App is not active
	998	Bad request / invalid params
499		Unexpected errors (Ref: xxxx)
RP01		App ID not matched • Request AppId not match with Recurrent Token's AppId
RP02		Insufficient balance
RP03		Exceeding account balance limit
RP04		Topup count exceeded for the user

8.9.2 Invalidate Token

Description:	Request to invalidate the recurrentToken
Request Method:	POST
URL:	https://{SERVER_DOMAIN}/paymentApi/payment/recurrent/token/invalidation

Request Parameters:

Parameter	***Mandatory***	***Example***	***Remarks***
appId	Y	70136705	App ID of the Merchant App
recurrentToken	Y	kis8lk8yt4wetez5mkd8l…	Token for recurrent payment
timestamp	Y	140235441215746	Unix Timestamp
sign	Y	dk8ued893j…	Signature

Response Fields:

Field	***Example***	***Remarks***
resultCode	0	Result code
message	Request Success	Result message
sign	sd9e4hf684…	Signature

Possible Result:

Result Code	***Reference Code***	***Description***
0		Request Success
400		Bad request • Mandatory parameters are not provided • Method not allowed
403		Forbidden • Incoming IP not registered
461		Invalid timestamp • The provided timestamp is outside of the valid API time period
462		Invalid Signature

		• Signature is not matched with the request parameters	
489		Invalid information • Cannot obtain apiKey to check signature	
491		Invalidate token failed (Ref: xxxx)	
	G01	Token not found	
	998	Bad request / invalid params	
	446	Service unavailable	
	447	Service unavailable	
499		Unexpected errors (Ref: xxxx)	

8.9.3 Query Payment Status

Description:	Request to query the payment status
Request Method:	POST
URL:	https://{SERVER_DOMAIN}/paymentApi/payment/status

Request Parameters:

Parameter	***Mandatory***	***Example***	***Remarks***
appId	Y	70136705	App ID of the Merchant App
merTradeNo	Y	12915236	A numbered transaction ID (Given by merchant. Unique per the merchant) Format: alphanumeric Max. length: 64
timestamp	Y	140235441215746	Unix Timestamp
sign	Y	dk8ued893j…	Signature

Response Fields:

Field	***Example***	***Remarks***
resultCode	0	Result code
message	Request Success	Result message
merTradeNo	12915236	A numbered transaction ID (Given by merchant. Unique per the merchant)
tradeNo	150423JHGgB	Tap & Go Transaction ID
tradeStatus	TRADE_FINISHED	Current status of this payment request. TRADE_FINISHED – payment success TRADE_CLOSED – payment cancelled / failed WAIT_TO_PAY – payment is

		processing
transactionType	CR	CR – Purchase (by Tap & Go user to merchant) DB – Top-up (from merchant to Tap & Go user)
sign	sd9e4hf684…	Signature

Possible Result:

Result Code	***Reference Code***	***Description***
0		Request Success
400		Bad request • Mandatory parameters are not provided • Method not allowed
403		Forbidden • Incoming IP not registered
461		Invalid timestamp • The provided timestamp is outside of the valid API time period
462		Invalid Signature • Signature is not matched with the request parameters
489		Invalid information • Cannot obtain apiKey to check signature
493		Query payment status failed (Ref: xxxx)
	ZB185	Illegal parameters
	ZB188	Service is not activated for this partner id
	444	Service unavailable
	445	Service unavailable
499		Unexpected errors (Ref: xxxx)

9 File Based Interface

9.1 Supported Transaction Type

1. File based protocol supports two types of transaction - Top-Up Transaction
2. The type of transaction will be indicated by file name.

9.2 Requirement

1. Supported file transfer protocol is SFTP only. Merchant is required to provide their SSH public key to Tap&Go Operation team for SFTP server configuration.
2. Request and Response files will be protected by GNU Privacy Guard (GnuPG). Tap&Go operation team and Merchant operation team will exchange their GnuPG public key.
3. For request file, Merchant is required to encrypt it with Tap&Go's GnuPG public key before sending out.
4. For response file, Tap&Go system will encrypt it with Merchant's GnuPG public key before sending out. Merchant can use their own private key to decrypt and validate.
5. Merchant servers which will initialize file transfer to Tap&Go are required to be registered and their IP should be fixed.
6. Request file is transferred to Tap&Go system during an agreed time slot (e.g. 01:00 a.m. to 03:00 a.m.) each day for the transaction requests of previous day.
7. Merchant is required to send request file to Tap&Go on every day, including nil request.

9.3 Production Server Configuration

Field	**Value**
Server Domain:	gateway2.tapngo.com.hk

9.4 UAT Server Configuration

Field	**Value**
Server Domain:	gateway2.sandbox.tapngo.com.hk

9.5 Keys Exchange

1. Merchant and Tap&Go Operation team will exchange the pubic keys by email and verbally confirmed by phone calls.
2. Development team will be assigned with different set of public keys and exchanged by emails.

9.6 File Format

9.6.1 Core Attributes

1. Content Encoding: UTF-8
2. Row terminator: CRLF (hex: ODOA)
3. Field delimiter: | (hex: 7c)
4. String enclosure: String in each field is enclosed by double quote character “ (hex: 22)

9.6.2 Top-Up Request File

File Name:

TAPNGO_[MerchantID]_YYYYMMDD_TOPUP_REQUEST.txt.gpg
where [MerchantID] is the assigned merchant ID in decimal format.

File Header Row:

FILEBEGIN<Transaction Date>

where <Transaction Date> is in format of YYYYDDMM

File Record:

File Number	Field Name	Field Format	Mandatory	Description
1	Token	ASCII String	Yes	ASCII string returned by Tap&Go Payment SDK
2	Amount	DDD.DD	Yes	Amount to be top-up to corresponding Tap&Go account
3	Remark	ASCII string	NO	Optional information of this top-up request.

File Footer Row

FILEEND<Total number of records in decimal number>

Sample 1:

Scenario	• On 1-Aug-2016, Merchant (ID=1088) approves 3 top-up requests. • Request file is generated at submit to Tap&Go system on 2-Aug after mid-night batch job.
File name	TAPNGO_1088_20160802_TOPUPREQUEST.txt.gpg
File content	FILEBEGIN20160801 fx3fe13r3fkxlh13lx31x34|100.00|dividend ghwe11fdf2xlh13lnfg08y|100.00|dividend bfk9qlkslajk3lxhl9081hlx|100.00|dividend FILEEND3

Sample 2:

Scenario	• On 2-Aug-2016, Merchant (ID=1088) approves nil top-up request. • Request file is generated at submit to Tap&Go system on 3-Aug after mid-night batch job.
File name	TAPNGO_1088_20160803_TOPUP_REQUEST.txt.gpg
File content	FILEBEGIN20160802 FILEEND0

9.6.3 Top-Up Response File

File Name:

TAPNGO_[MerchantID]_YYYYMMDD_TOPUP_RESPONSE.txt.gpg

where [MerchantID] is the assigned merchant ID in decimal format.

File Header Row:

FILEBEGIN<Transaction Date>

where <Transaction Date> is in format of YYYYDDMM

File Record:

Field Name	**Field Format**	**Mandatory**	**Description**
Token	ASCII String	Yes	ASCII string returned by Tap&Go Payment SDK
Amount	DDDDD.DD	Yes	Amount to be top-up to corresponding Tap&Go account
Result	DDD	YES	Result Code in 3 decimal digits. 000 = success 001 = Failed

File Footer Row:

FILEEND<Total number of records in decimal number>

Sample 1:

Scenario	• On 2-Aug-2016, Tap&Go system complete process 1 top-up request file from Merchant ID=1088.
File name	TAPNGO_1088_20160802_TOPUP_RESPONSE.txt
File content	FILEBEGIN20160801 fx3fe13r3fkxlh13lx31x34|100.00|dividend ghwe11fdf2xlh13lnfg08y|100.00|dividend bfk9qlkslajk3lxhl9081hlx|100.00|dividend FILEEND3

10 Appendix

10.1 Web Payment Gateway error codes

Error Code	**Description**
AP001	Internal server error
AP002	Merchant info. invalid or not found
AP003	Signature check failed
AP004	Internal server error
AP005	Payload error
AP010	Payment session timeout
AP011	Fail to receive payment status update
AP998	Invalid request parameters
AP999	Unknown error
0499	Extras field checking failed

10.2 SDK error codes

Error Code	**Error message**	**Description**
SS100	Merchant enter App ID format incorrect. (SS100)	Merchant enter appId format incorrect
SS101	Merchant enter API Key format incorrect. (SS101)	Merchant enter apiKey format incorrect
SS102	Merchant enter Public Key format incorrect. (SS102)	Merchant enter publicKey format incorrect
SS103	Merchant enter Merchant trade number format incorrect. (SS103)	Merchant enter merTradeNo format incorrect
SS104	Merchant enter Total price format incorrect. (SS104)	Merchant enter totalPrice format incorrect

SS105	Merchant enter Current format incorrect. (SS105)	Merchant enter currenct format incorrect
SS106	Merchant enter Remark format incorrect. (SS106)	Merchant enter remark format incorrect
SS107	Merchant enter Payment type format incorrect. (SS107)	Merchant enter paymentType format incorrect
SS108	Merchant enter Extra format incorrect. (SS108)	Merchant enter extra format incorrect
SS109	Merchant enter Notify URL format incorrect (SS109)	Merchant enter notifyUrl format incorrect
SS110	Merchant enter Call Back ID format incorrect (SS110)	Merchant enter Call Back ID format incorrect
SS200	SDK cannot find Tap&Go App or the Tap&Go App does not merchant payment. (SS200)	SDK cannot find Tap&Go App or the Tap&Go App does not support P2M
SS300	Payment result error. (SS300)	Payment response error
SS301	Payment result error. (SS301)	Payment response error
SS400	Request timeout. (SS400)	Request timeout
SS500	Payment not set. (SS500)	Payment not set
SS999	Unknown error (SS999)	Unknown error
SM100	Payment authentication fail. (SM100)	Payment authentication fail
SM101	Payment authentication fail. (SM101)	Payment authentication fail
SM102	Payment authentication fail. (SM102)	Payment authentication fail
SM103	The device time is not allow for payment. (SM103)	The device time is not allow for the SDK
SM999	Unknown error. (SM999)	Unknown error
SA001	User cancel the payment (SA100)	User cancel the payment
SA100	Payment request invalid (SA100)	Payment request invalid
SA101	Payment request invalid (SA101)	Payment request invalid
SA102	Payment request invalid (SA102)	Payment request invalid
SA103	Payment request invalid (SA103)	Payment request invalid
SA104	Payment request invalid (SA104)	Payment request invalid
SA105	Payment request invalid (SA105)	Payment request invalid
SA106	Payment request invalid (SA106)	Payment request invalid

SA107	Payment request invalid (SA107)	Payment request invalid
SA200	Payment decline(Tap&Go try to update) (SA200)	Payment decline(Tap&Go try to update)
SA201	Payment decline(Tap&Go try to update) (SA201)	Payment decline(Tap&Go try to update)
SA300	Payment decline(Tap&Go error) (SA300)	Payment decline(Tap&Go error)
SA400	Payment decline(Tap&Go error) (SA400)	Payment decline(Tap&Go error)
SA401	Payment decline(Tap&Go error) (SA401)	Payment decline(Tap&Go error)
SA402	Payment decline(Tap&Go error) (SA402)	Payment decline(Tap&Go error)
SA403	Payment decline(Tap&Go error) (SA403)	Payment decline(Tap&Go error)
SA404	Payment decline(Tap&Go error) (SA404)	Payment decline(Tap&Go error)
SA500	Payment decline(Tap&Go is not registered) (SA500)	Payment decline(Tap&Go is not registered)
SA501	Payment decline(Tap&Go is not registered) (SA501)	Payment decline(Tap&Go is not registered)
SA502	Payment decline(Tap&Go is not registered) (SA502)	Payment decline(Tap&Go is not registered)
SA503	Payment decline(Tap&Go is not registered) (SA503)	Payment decline(Tap&Go is not registered)
SA504	Payment decline(Tap&Go is not registered) (SA504)	Payment decline(Tap&Go is not registered)
SA505	Payment decline(Tap&Go is not registered) (SA505)	Payment decline(Tap&Go is not registered)
SA998	Request cancel(Android only) (SA998)	Request cancel(Android only)
SA999	Unknown error (SA999)	Unknown error
SP100	Payment processing error (SP100)	Payment processing error
SP101	Payment processing error (SP101)	Payment processing error
SP102	Payment processing error (SP102)	Payment processing error
SP200	Payment fail (SP200)	Payment fail
SP201	Payment fail (SP201)	Payment fail
SP202	Payment fail (SP202)	Payment fail
SP999	Unknown error (SP999)	Unknown error

- END -