

WePay iOS SDK

2.0.0

Generated by Doxygen 1.8.8

Mon Jul 6 2015 17:12:52

Contents

1	Main Page	1
2	Hierarchical Index	6
2.1	Class Hierarchy	6
3	Class Index	7
3.1	Class List	7
4	File Index	7
4.1	File List	7
5	Class Documentation	7
5.1	WePay Class Reference	7
5.1.1	Detailed Description	8
5.1.2	Method Documentation	8
5.1.3	Property Documentation	10
5.2	WPAAddress Class Reference	10
5.2.1	Detailed Description	11
5.2.2	Method Documentation	11
5.2.3	Property Documentation	12
5.3	<WPCardReaderDelegate> Protocol Reference	13
5.3.1	Detailed Description	13
5.3.2	Method Documentation	13
5.4	<WPCheckoutDelegate> Protocol Reference	14
5.4.1	Detailed Description	14
5.4.2	Method Documentation	14
5.5	WPConfig Class Reference	15
5.5.1	Detailed Description	15
5.5.2	Method Documentation	15
5.5.3	Property Documentation	16
5.6	WPPaymentInfo Class Reference	17
5.6.1	Detailed Description	17
5.6.2	Method Documentation	17
5.6.3	Property Documentation	18
5.7	WPPaymentToken Class Reference	19
5.7.1	Detailed Description	19
5.7.2	Method Documentation	19

5.7.3	Property Documentation	20
5.8	<WPTokenizationDelegate> Protocol Reference	20
5.8.1	Detailed Description	20
5.8.2	Method Documentation	20
6	File Documentation	21
6.1	/Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/WPErrors.h File Reference	21
6.1.1	Detailed Description	22
6.1.2	Macro Definition Documentation	22
6.1.3	Enumeration Type Documentation	23
6.1.4	Variable Documentation	23
Index		25

1 Main Page

Introduction

This iOS SDK enables collection of payments via various payment methods(described below).

It is meant for consumption by [WePay](#) partners who are developing their own iOS apps aimed at merchants and/or consumers.

Regardless of the payment method used, the SDK will ultimately return a Payment Token, which must be redeemed via a server-to-server API call to complete the transaction.

Payment methods

There are two types of payment methods:

- Consumer payment methods - to be used in apps where consumers directly pay and/or make donations
- Merchant payment methods - to be used in apps where merchants collect payments from their customers

The [WePay](#) iOS SDK supports the following payment methods

- Card Swiper (Merchant) Using a Card Swiper, a merchant can accept in-person payments by swiping a consumer's card.
- Manual Entry (Consumer/Merchant) The Manual Entry payment method lets consumer and merchant apps accept payments by allowing the user to manually enter card info.

Installation

- In Xcode, go to your app's target settings. On the Build Phases tab, expand the Link Binary With Libraries section.
- Include the following iOS frameworks:
 - AudioToolbox.framework

- AVFoundation.framework
 - CoreBluetooth.framework
 - CoreTelephony.framework
 - MediaPlayer.framework
 - SystemConfiguration.framework
 - libstdc++.6.0.9.dylib
- Also include the following frameworks provided with this SDK:
 - G4XSwiper.framework
 - RPx.framework
 - RUA.framework
 - TrustDefenderMobile.framework
 - WePay.framework
- Done!

SDK Organization

[WePay.h](#)

[WePay.h](#) is the starting point for consuming the SDK, and contains the primary class you will interact with. It exposes all the methods you can call to accept payments via the supported payment methods. Detailed reference documentation is available on the reference page for the [WePay](#) class.

Delegate protocols

The SDK uses delegate protocols to respond to API calls. You must adopt the relevant protocols to receive responses to the API calls you make. Detailed reference documentation is available on the reference page for each protocol:

- [WPCardReaderDelegate](#)
- [WPCheckoutDelegate](#)
- [WPTokenizationDelegate](#)

Data Models

All other classes in the SDK are data models that are used to exchange data between your app and the SDK. Detailed reference documentation is available on the reference page for each class.

Next Steps

Head over to the [WePay](#) class reference to see all the API methods available. When you are ready, look at the samples below to learn how to interact with the SDK.

Error Handling

[WPError.h](#) serves as documentation for all errors surfaced by the [WePay](#) iOS SDK.

Samples

See the [WePayExample](#) app for a working implementation of all API methods.

Initializing the SDK

- Complete the installation steps (above).
- Include [WePay.h](#)

```
#import <WePay/WePay.h>
```

- Define a property to store the [WePay](#) object

```
\@property (nonatomic, strong) WePay *wepay;
```

- Create a [WPConfig](#) object

```
WPConfig *config = [[WPConfig alloc] initWithClientId:@"your_client_id" environment:
    kWPEnvironmentStage];
```

- Initialize the [WePay](#) object and assign it to the property

```
self.wepay = [[WePay alloc] initWithConfig:config];
```

(optional) Providing permission to use location services for fraud detection

- In Xcode, go to your projects settings. On the Build Phases tab, expand the Link Binary With Libraries section and include the CoreLocation.framework iOS framework.
- Open your app's Info.plist file and add entries for `NSLocationUsageDescription` and `NSLocationWhenInUseUsageDescription`.

```
1 <key>NSLocationUsageDescription</key>
2 <string>Location information is used for fraud prevention</string>
3 <key>NSLocationWhenInUseUsageDescription</key>
4 <string>Location information is used for fraud prevention</string>
```

- Set the option on the config object, before initializing the [WePay](#) object

```
config.useLocation = YES;
```

Integrating the Swipe payment method

- Adopt the [WPCardReaderDelegate](#) and [WPTokenizationDelegate](#) protocols

```
\@interface MyWePayDelegate : NSObject <WPCardReaderDelegate,
    WPTokenizationDelegate>
```

- Implement the [WPCardReaderDelegate](#) protocol methods

```
- (void) cardReaderDidChangeStatus:(id) status
{
    if (status == kWPCardReaderStatusNotConnected) {
        // show UI that prompts the user to connect the Card Reader
        self.statusLabel.text = @"Connect Card Reader";
    } else if (status == kWPCardReaderStatusWaitingForSwipe) {
        // show UI that prompts the user to swipe
        self.statusLabel.text = @"Swipe Card";
    } else if (status == kWPCardReaderStatusSwipeDetected) {
        // provide feedback to the user that a swipe was detected
        self.statusLabel.text = @"Swipe Detected...";
    }
}
```

```

    } else if (status == kWPCardReaderStatusTokenizing) {
        // provide feedback to the user that the card info is being tokenized/verified
        self.statusLabel.text = @"Tokenizing...";
    } else if (status == kWPCardReaderStatusStopped) {
        // provide feedback to the user that the swiper has stopped
        self.statusLabel.text = @"Card Reader Stopped";
    }
}

- (void) didReadPaymentInfo:(WPPaymentInfo *)paymentInfo
{
    // use the payment info (for display/recordkeeping)
    // wait for tokenization response
}

- (void) didFailToReadPaymentInfoWithError:(NSError *)error
{
    // Handle the error
}

```

- Implement the [WPTokenizationDelegate](#) protocol methods

```

- (void) paymentInfo:(WPPaymentInfo *)paymentInfo didTokenize:(
    WPPaymentToken *)paymentToken
{
    // Send the tokenId (paymentToken.tokenId) to your server
    // Your server would use the tokenId to make a /checkout/create call to complete the transaction
}

- (void) paymentInfo:(WPPaymentInfo *)paymentInfo didFailTokenization:(NSError *)error
{
    // Handle the error
}

```

- Make the [WePay](#) API call, passing in the instance(s) of the class(es) that implemented the delegate protocols

```

[self.wepay startCardReaderForTokenizingWithCardReaderDelegate:self tokenizationDelegate:self];
// Show UI asking the user to insert the card reader and wait for it to be ready

```

- That's it! The following sequence of events will occur:

1. The user inserts the card reader (or it is already inserted)
2. The SDK tries to detect the card reader and initialize it.
 - If the card reader is not detected, the `cardReaderDidChangeStatus:` method will be called with `status = kWPCardReaderStatusNotConnected`
 - If the card reader is successfully detected, then the `cardReaderDidChangeStatus:` method will be called with `status = kWPCardReaderStatusConnected`.
 - Next, if the card reader is successfully initialized, then the `cardReaderDidChangeStatus:` method will be called with `status = kWPCardReaderStatusWaitingForSwipe`
 - Otherwise, `didFailToReadPaymentInfoWithError:` will be called with the appropriate error, and processing will stop (the `cardReaderDidChangeStatus:` method will be called with `status = kWPCardReaderStatusStopped`)
3. If the card reader successfully initialized, it will wait for the user to swipe a card
4. If a recoverable error occurs during swiping, the `didFailToReadPaymentInfoWithError:` method will be called. After a few seconds, the `cardReaderDidChangeStatus:` method will be called with `status = kWPCardReaderStatusWaitingForSwipe` and the card reader will again wait for the user to swipe a card
5. If an unrecoverable error occurs during swiping, or the user does not swipe, the `didFailToReadPaymentInfoWithError:` method will be called, and processing will stop
6. Otherwise, the user swiped successfully, and the `cardReaderDidChangeStatus:` method will be called with `status = kWPCardReaderStatusSwipeDetected` followed by the `didReadPaymentInfo:` method

7. Next, the SDK will automatically send the obtained card info to [WePay](#)'s servers for tokenization (the `cardReaderDidChangeStatus:` method will be called with `status = kWPCardReaderStatusTokenizing`)
8. If the tokenization succeeds, the `paymentInfo:didTokenize:` method will be called
9. Otherwise, if the tokenization fails, the `paymentInfo:didFailTokenization:` method will be called with the appropriate error

Integrating the Manual payment method

- Adopt the [WPTokenizationDelegate](#) protocol

```
\@interface MyWePayDelegate : NSObject <WPTokenizationDelegate>
```

- Implement the [WPTokenizationDelegate](#) protocol methods

```
- (void) paymentInfo:(WPPaymentInfo *)paymentInfo didTokenize:(
    WPPaymentToken *)paymentToken
{
    // Send the tokenId (paymentToken.tokenId) to your server
    // Your server can use the tokenId to make a /checkout/create call to complete the transaction
}

- (void) paymentInfo:(WPPaymentInfo *)paymentInfo didFailTokenization:(NSError *)error
{
    // Handle error
}
```

- Instantiate a [WPPaymentInfo](#) object using the user's credit card and address data

```
WPPaymentInfo *paymentInfo = [[WPPaymentInfo alloc] initWithFirstName:@"WPiOS"
                                                                    lastName:@"Example"
                                                                    email:@"wp.ios.example@wepay.com"
                                                                    billingAddress:[
                                                                    WPAddress alloc] initWithZip:@"94306"]
                                                                    shippingAddress:nil
                                                                    cardNumber:@"5496198584584769"
                                                                    cvv:@"123"
                                                                    expMonth:@"04"
                                                                    expYear:@"2020"
                                                                    virtualTerminal:YES];
// Note: the virtualTerminal parameter above should be set to YES if a merchant is collecting payments
// manually using your app. It should be set to NO if a payer is making a manual payment using your app.
```

- Make the [WePay](#) API call, passing in the instance of the class that implemented the [WPTokenizationDelegate](#) protocol

```
[self.wepay tokenizeManualPaymentInfo:paymentInfo tokenizationDelegate:self];
```

- That's it! The following sequence of events will occur:
 1. The SDK will send the obtained payment info to [WePay](#)'s servers for tokenization
 2. If the tokenization succeeds, the `paymentInfo:didTokenize:` method will be called
 3. Otherwise, if the tokenization fails, the `paymentInfo:didFailTokenization:` method will be called with the appropriate error

Integrating the Store Signature API

- Adopt the [WPCheckoutDelegate](#) protocol

```
\@interface MyWePayDelegate : NSObject <WPCheckoutDelegate>
```

- Implement the [WPCheckoutDelegate](#) protocol methods

```

- (void) didStoreSignature:(NSString *)signatureUrl
    forCheckoutId:(NSString *)checkoutId
{
    // success! nothing to do here
}

- (void) didFailToStoreSignatureImage:(UIImage *)image
    forCheckoutId:(NSString *)checkoutId
    withError:(NSError *)error
{
    // handle the error
}

```

- Obtain the `checkout_id` associated with this signature from your server

```
NSString *checkoutId = [self obtainCheckoutId];
```

- Instantiate a `UIImage` containing the user's signature

```
UIImage *signature = [UIImage imageNamed:@"dd_signature.png"];
```

- Make the [WePay](#) API call, passing in the instance of the class that implemented the [WPCheckoutDelegate](#) protocol

```
[self.wepay storeSignatureImage:signature
    forCheckoutId:checkoutId
    checkoutDelegate:self];
```

- That's it! The following sequence of events will occur:
 1. The SDK will send the obtained signature to [WePay's](#) servers
 2. If the operation succeeds, the `didStoreSignature:forCheckoutId:` method will be called
 3. Otherwise, if the operation fails, the `didFailToStoreSignatureImage:forCheckoutId:withError:` method will be called with the appropriate error

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<NSObject>

WePay	7
WPAddress	10
<WPCardReaderDelegate>	13
<WPCheckoutDelegate>	14
WPConfig	15
WPPaymentInfo	17
WPPaymentToken	19
<WPTokenizationDelegate>	20

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

WePay	7
WPAddress	10
<WPCardReaderDelegate>	13
<WPCheckoutDelegate>	14
WPConfig	15
WPPaymentInfo	17
WPPaymentToken	19
<WPTokenizationDelegate>	20

4 File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

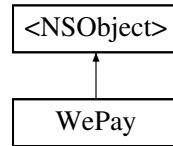
/Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/WePay.h	??
/Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/WPError.h WPError.h serves as documentation for all errors surfaced by the WePay iOS SDK	21
/Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/Models/WPAddress.h	??
/Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/Models/WPConfig.h	??
/Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/Models/WPPaymentInfo.h	??
/Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/Models/WPPaymentToken.h	??

5 Class Documentation

5.1 WePay Class Reference

```
#import <WePay.h>
```

Inheritance diagram for WePay:



Instance Methods

Initialization

- (instancetype) - [initWithConfig:](#)

Tokenization

- (void) - [tokenizePaymentInfo:tokenizationDelegate:](#)

Card Reader related methods

- (void) - [startCardReaderForReadingWithCardReaderDelegate:](#)
- (void) - [startCardReaderForTokenizingWithCardReaderDelegate:tokenizationDelegate:](#)
- (void) - [stopCardReader](#)

Checkout related methods

- (void) - [storeSignatureImage:forCheckoutId:checkoutDelegate:](#)

Properties

- [WPConfig](#) * [config](#)

5.1.1 Detailed Description

Main Class containing all public endpoints.

5.1.2 Method Documentation

5.1.2.1 - (instancetype) initWithConfig: (WPConfig *) config

The designated initializer. Use this to initialize the SDK.

Parameters

<i>config</i>	A WPConfig instance.
---------------	--------------------------------------

Returns

A [WePay](#) instance, which can be used to access most of the functionality of this sdk.

5.1.2.2 - (void) startCardReaderForReadingWithCardReaderDelegate: (id< WPCardReaderDelegate >) cardReaderDelegate

Initializes the card reader for reading card info.

The card reader will wait 60 seconds for a card, and then return a timeout error if a card is not detected. The card reader will automatically stop waiting for card if:

- a timeout occurs
- a card is successfully detected
- an unexpected error occurs
- stopCardReader is called

However, if a general error (domain:kWPErrCategoryCardReader, errorCode:WPErrorCardReaderGeneralError) occurs while reading, after a few seconds delay, the card reader will automatically start waiting again for another 60 seconds. At that time, [WPCardReaderDelegate](#)'s cardReaderDidChangeStatus: method will be called with kWPCardReaderStatusWaitingForSwipe, and the user can try to use the card reader again. This behavior can be configured with [WPConfig](#).

WARNING: When this method is called, a (normally inaudible) signal is sent to the headphone jack of the phone, where the card reader is expected to be connected. If headphones are connected instead of the card reader, they may emit a very loud audible tone on receiving this signal. This method should only be called when the user intends to use the card reader.

Parameters

<i>cardReaderDelegate</i>	The delegate class which will receive the response(s) for this call.
---------------------------	--

5.1.2.3 - (void) startCardReaderForTokenizingWithCardReaderDelegate: (id< WPCardReaderDelegate >) cardReaderDelegate tokenizationDelegate:(id< WPTokenizationDelegate >) tokenizationDelegate

The card reader will wait 60 seconds for a card, and then return a timeout error if a card is not detected. The card reader will automatically stop waiting for card if:

- a timeout occurs
- a card is successfully detected
- an unexpected error occurs
- stopCardReader is called

However, if a general error (domain:kWPErrCategoryCardReader, errorCode:WPErrorCardReaderGeneralError) occurs while reading, after a few seconds delay, the card reader will automatically start waiting again for another 60 seconds. At that time, [WPCardReaderDelegate](#)'s cardReaderDidChangeStatus: method will be called with kWPCardReaderStatusWaitingForSwipe, and the user can try to use the card reader again. This behavior can be configured with [WPConfig](#).

WARNING: When this method is called, a (normally inaudible) signal is sent to the headphone jack of the phone, where the card reader is expected to be connected. If headphones are connected instead of the card reader, they may emit a very loud audible tone on receiving this signal. This method should only be called when the user intends to use the card reader.

Parameters

<i>cardReader↔ Delegate</i>	The delegate class which will receive the response(s) for this call.
<i>tokenization↔ Delegate</i>	The delegate class which will receive the tokenization response(s) for this call.

5.1.2.4 - (void) stopCardReader

Stops the card reader. In response, [WPCardReaderDelegate](#)'s `cardReaderDidChangeStatus:` method will be called with `kWPCardReaderStatusStopped`. Any tokenization in progress will not be stopped, and its result will be delivered to the [WPTokenizationDelegate](#).

5.1.2.5 - (void) storeSignatureImage: (UIImage *) image forCheckoutId:(NSString *) checkoutId checkoutDelegate:(id< WPCheckoutDelegate >) checkoutDelegate

Stores a signature image associated with a checkout id on [WePay](#)'s servers. The signature can be retrieved via a server-to-server call that fetches the checkout object. The aspect ratio (width:height) of the image must be between 1:4 and 4:1. If needed, the image will internally be scaled to fit inside 256x256 pixels, while maintaining the original aspect ratio.

Parameters

<i>image</i>	The signature image to be stored.
<i>checkoutId</i>	The checkout id associated with this transaction.
<i>checkout↔ Delegate</i>	The delegate class which will receive the response(s) for this call.

5.1.2.6 - (void) tokenizePaymentInfo: (WPPaymentInfo *) paymentInfo tokenizationDelegate:(id< WPTokenizationDelegate >) tokenizationDelegate

Creates a payment token from a [WPPaymentInfo](#) object.

Parameters

<i>paymentInfo</i>	The payment info obtained from the user in any form.
<i>tokenization↔ Delegate</i>	The delegate class which will receive the tokenization response(s) for this call.

5.1.3 Property Documentation

5.1.3.1 -(WPConfig*) config [read],[nonatomic],[strong]

Your [WePay](#) config

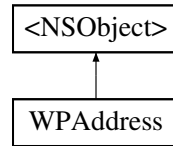
The documentation for this class was generated from the following file:

- /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/WePay.h

5.2 WPAAddress Class Reference

```
#import <WPAAddress.h>
```

Inheritance diagram for WPAAddress:



Instance Methods

- (instancetype) - [initWithZip:](#)
- (instancetype) - [initWithAddress1:address2:city:state:zip:](#)
- (instancetype) - [initWithAddress1:address2:city:region:postcode:country:](#)
- (NSDictionary *) - **toDict**

Properties

- NSString * [address1](#)
- NSString * [address2](#)
- NSString * [city](#)
- NSString * [country](#)
- NSString * [postcode](#)
- NSString * [region](#)
- NSString * [state](#)
- NSString * [zip](#)

5.2.1 Detailed Description

An instance of this class represents a physical address.

5.2.2 Method Documentation

5.2.2.1 - (instancetype) initWithAddress1: (NSString *) *address1* address2:(NSString *) *address2* city:(NSString *) *city* region:(NSString *) *region* postcode:(NSString *) *postcode* country:(NSString *) *country*

Initializes a non-US Address.

Parameters

<i>address1</i>	The first line of the street address.
<i>address2</i>	The second line of the street address.
<i>city</i>	The city.
<i>region</i>	The region. Only for non-US addresses when available.
<i>postcode</i>	The postcode. Only for non-US addresses when available.
<i>country</i>	The 2-letters ISO-3166-1 country code.

Returns

The address.

5.2.2.2 - (instancetype) initWithAddress1: (NSString *) *address1* address2:(NSString *) *address2* city:(NSString *) *city* state:(NSString *) *state* zip:(NSString *) *zip*

Initializes a US Address.

Parameters

<i>address1</i>	The first line of the street address.
<i>address2</i>	The second line of the street address.
<i>city</i>	The city.
<i>state</i>	The 2-letters US state code.
<i>zip</i>	The US zip or zip-plus code.

Returns

The address.

5.2.2.3 - (instancetype) initWithZip: (NSString *) zip

Initializes a US Address with just a zip.

Parameters

<i>zip</i>	The US zip or zip-plus code.
------------	------------------------------

Returns

The address.

5.2.3 Property Documentation**5.2.3.1 - (NSString*) address1 [read], [nonatomic], [strong]**

The first line of the street address.

5.2.3.2 - (NSString*) address2 [read], [nonatomic], [strong]

The second line of the street address.

5.2.3.3 - (NSString*) city [read], [nonatomic], [strong]

The city.

5.2.3.4 - (NSString*) country [read], [nonatomic], [strong]

The 2-letters ISO-3166-1 country code.

5.2.3.5 - (NSString*) postcode [read], [nonatomic], [strong]

The postcode. Only for non-US addresses when available.

5.2.3.6 - (NSString*) region [read], [nonatomic], [strong]

The region. Only for non-US addresses when available.

5.2.3.7 - (NSString*) state [read], [nonatomic], [strong]

The 2-letters US state code. Only for US addresses.

5.2.3.8 - (NSString*) zip [read], [nonatomic], [strong]

The US zip or zip-plus code. Only for US addresses.

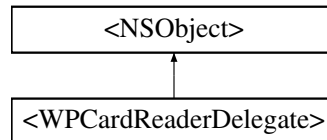
The documentation for this class was generated from the following file:

- /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/Models/WPAddress.h

5.3 <WPCardReaderDelegate> Protocol Reference

```
#import <WePay.h>
```

Inheritance diagram for <WPCardReaderDelegate>:



Instance Methods

- (void) - [didReadPaymentInfo:](#)
- (void) - [didFailToReadPaymentInfoWithError:](#)
- (void) - [cardReaderDidChangeStatus:](#)

5.3.1 Detailed Description

This delegate protocol has to be adopted by any class that handles Card Reader responses.

5.3.2 Method Documentation

5.3.2.1 - (void) cardReaderDidChangeStatus: (id) *status*

Called when the card reader changes status.

Parameters

<i>status</i>	Current status of the card reader, one of (kWPCardReaderStatusNotConnected, kWPCardReaderStatusConnected, kWPCardReaderStatusWaitingForSwipe, kWPCardReaderStatusSwipeDetected, kWPCardReaderStatusTokenizing, kWPCardReaderStatusStopped).
---------------	---

5.3.2.2 - (void) didFailToReadPaymentInfoWithError: (NSError *) *error*

Called when an error occurs while reading a card.

Parameters

<i>error</i>	The error which caused the failure.
--------------	-------------------------------------

5.3.2.3 - (void) didReadPaymentInfo: (WPPaymentInfo *) *paymentInfo*

Called when payment info is successfully obtained from a card.

Parameters

<i>paymentInfo</i>	The payment info.
--------------------	-------------------

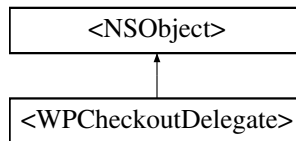
The documentation for this protocol was generated from the following file:

- /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/WePay.h

5.4 <WPCheckoutDelegate> Protocol Reference

```
#import <WePay.h>
```

Inheritance diagram for <WPCheckoutDelegate>:

**Instance Methods**

- (void) - [didStoreSignature:forCheckoutId:](#)
- (void) - [didFailToStoreSignatureImage:forCheckoutId:withError:](#)

5.4.1 Detailed Description

This delegate protocol has to be adopted by any class that handles Checkout responses.

5.4.2 Method Documentation**5.4.2.1 - (void) didFailToStoreSignatureImage: (UIImage *) image forCheckoutId:(NSString *) checkoutId withError:(NSError *) error**

Called when an error occurs while storing a signature.

Parameters

<i>image</i>	The signature image to be stored.
<i>checkoutId</i>	The checkout id associated with the signature.
<i>error</i>	The error which caused the failure.

5.4.2.2 - (void) didStoreSignature: (NSString *) signatureUrl forCheckoutId:(NSString *) checkoutId

Called when a signature is successfully stored for the given checkout id.

Parameters

<i>signatureUrl</i>	The url for the signature image.
---------------------	----------------------------------

<i>checkoutId</i>	The checkout id associated with the signature.
-------------------	--

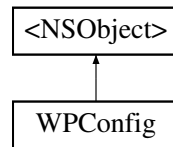
The documentation for this protocol was generated from the following file:

- /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/WePay.h

5.5 WPConfig Class Reference

```
#import <WPConfig.h>
```

Inheritance diagram for WPConfig:



Instance Methods

- (instancetype) - [initWithClientId:environment:](#)
- (instancetype) - [initWithClientId:environment:useLocation:restartCardReaderAfterSuccess:restartCardReaderAfterGeneralError:restartCardReaderAfterOtherErrors:](#)

Properties

- NSString * [clientId](#)
- NSString * [environment](#)
- BOOL [useLocation](#)
- BOOL [restartCardReaderAfterSuccess](#)
- BOOL [restartCardReaderAfterGeneralError](#)
- BOOL [restartCardReaderAfterOtherErrors](#)

5.5.1 Detailed Description

The configuration object used for initializing a [WePay](#) instance.

5.5.2 Method Documentation

5.5.2.1 - (instancetype) [initWithClientId:](#) (NSString *) *clientId* [environment:](#)(NSString *) *environment*

A convenience initializer

Parameters

<i>clientId</i>	Your WePay clientId.
-----------------	--------------------------------------

<i>environment</i>	The environment to be used, one of (kWPEnvironmentStage, kWPEnvironmentProduction).
--------------------	---

Returns

A [WPConfig](#) instance which can be used to initialize a [WePay](#) instance.

5.5.2.2 - (instancetype) initWithClientId: (NSString *) *clientId* environment:(NSString *) *environment* useLocation:(BOOL) *useLocation* restartCardReaderAfterSuccess:(BOOL) *restartCardReaderAfterSuccess* restartCardReaderAfterGeneralError:(BOOL) *restartCardReaderAfterGeneralError* restartCardReaderAfterOtherErrors:(BOOL) *restartCardReaderAfterOtherErrors*

The designated initializer

Parameters

<i>clientId</i>	Your WePay clientId.
<i>environment</i>	The environment to be used, one of (kWPEnvironmentStage, kWPEnvironmentProduction).
<i>useLocation</i>	Flag to determine if we should use location services.
<i>restartCardReaderAfterSuccess</i>	Flag to determine if the card reader should automatically restart after a successful read.
<i>restartCardReaderAfterGeneralError</i>	Flag to determine if the card reader should automatically restart after a general error (domain: kWPErrorCategoryCardReader, errorCode: WPErrorCardReaderGeneralError).
<i>restartCardReaderAfterOtherErrors</i>	Flag to determine if the card reader should automatically restart after an error other than general error.

Returns

A [WPConfig](#) instance which can be used to initialize a [WePay](#) instance.

5.5.3 Property Documentation

5.5.3.1 - (NSString*) *clientId* [read], [nonatomic], [strong]

Your [WePay](#) clientId for the specified environment

5.5.3.2 - (NSString*) *environment* [read], [nonatomic], [strong]

The environment to be used, one of (staging, production)

5.5.3.3 - (BOOL) *restartCardReaderAfterGeneralError* [read], [write], [nonatomic], [assign]

Determines if the card reader should automatically restart after a general error (domain: kWPErrorCategoryCardReader, errorCode: WPErrorCardReaderGeneralError). Defaults to YES.

5.5.3.4 - (BOOL) *restartCardReaderAfterOtherErrors* [read], [write], [nonatomic], [assign]

Determines if the card reader should automatically restart after an error other than general error. Defaults to NO.

5.5.3.5 - (BOOL) *restartCardReaderAfterSuccess* [read], [write], [nonatomic], [assign]

Determines if the card reader should automatically restart after a successful read. Defaults to NO.

5.5.3.6 **-(BOOL) useLocation** [read], [write], [nonatomic], [assign]

Determines if we should use location services. Defaults to NO.

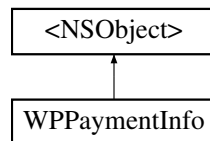
The documentation for this class was generated from the following file:

- /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/Models/WPConfig.h

5.6 WPPaymentInfo Class Reference

```
#import <WPPaymentInfo.h>
```

Inheritance diagram for WPPaymentInfo:



Instance Methods

- (instancetype) - **initWithSwipedInfo:**
- (instancetype) - **initWithFirstName:lastName:email:billingAddress:shippingAddress:cardNumber:cvv:expMonth:expYear:virtualTerminal:**
- (void) - **addEmail:**

Properties

- NSString * **firstName**
- NSString * **lastName**
- NSString * **email**
- NSString * **paymentDescription**
- BOOL **isVirtualTerminal**
- WPAddress * **billingAddress**
- WPAddress * **shippingAddress**
- id **paymentMethod**
- id **swiperInfo**
- id **manualInfo**

5.6.1 Detailed Description

An instance of this class represents the payment information obtained from the user via any of the supported payment methods. It is used as input for tokenization operations.

5.6.2 Method Documentation

5.6.2.1 **-(void) addEmail: (NSString *) email**

Allows adding an email if one is not already present. The call will be ignored if an email is already present.

Parameters

<i>email</i>	the email address to be added
--------------	-------------------------------

5.6.2.2 - (instancetype) initWithFirstName: (NSString *) *firstName* lastName:(NSString *) *lastName* email:(NSString *) *email* billingAddress:(WPAddress *) *billingAddress* shippingAddress:(WPAddress *) *shippingAddress* cardNumber:(NSString *) *cardNumber* cvv:(NSString *) *cvv* expMonth:(NSString *) *expMonth* expYear:(NSString *) *expYear* virtualTerminal:(BOOL) *virtualTerminal*

Initializes a [WPPaymentInfo](#) instance of type kWPPaymentMethodManual.

Parameters

<i>firstName</i>	First name of the payer.
<i>lastName</i>	Last name of the payer.
<i>email</i>	Email address of the payer.
<i>billingAddress</i>	Billing address.
<i>shippingAddress</i>	Shipping address.
<i>cardNumber</i>	The card number.
<i>cvv</i>	The cvv code.
<i>expMonth</i>	The 2-digit expiration month on the credit card.
<i>expYear</i>	The 4-digit expiration year on the credit card.
<i>virtualTerminal</i>	The virtual terminal flag - should be false if payment info was collected on the payer's device.

Returns

A [WPPaymentInfo](#) object initialized with manually obtained card info.

5.6.3 Property Documentation

5.6.3.1 - (WPAddress*) *billingAddress* [read], [nonatomic], [strong]

Billing address.

5.6.3.2 - (NSString*) *email* [read], [nonatomic], [strong]

Email address of the payer.

5.6.3.3 - (NSString*) *firstName* [read], [nonatomic], [strong]

First name of the payer.

5.6.3.4 - (BOOL) *isVirtualTerminal* [read], [nonatomic], [assign]

Determines if the card was obtained in virtual terminal mode.

5.6.3.5 - (NSString*) *lastName* [read], [nonatomic], [strong]

Last name of the payer.

5.6.3.6 - (id) *manualInfo* [read], [nonatomic], [strong]

Additional info obtained by using the Manual payment method.

5.6.3.7 - (NSString*) paymentDescription [read],[nonatomic],[strong]

Masked representation of the payment instrument. e.g. 4242XXXXXXXX1234

5.6.3.8 - (id) paymentMethod [read],[nonatomic],[strong]

The payment method used, one of (kWPPaymentMethodManual, kWPPaymentMethodSwipe).

5.6.3.9 - (WPAddress*) shippingAddress [read],[nonatomic],[strong]

Shipping address.

5.6.3.10 - (id) swiperInfo [read],[nonatomic],[strong]

Additional info obtained by using the Swipe payment method.

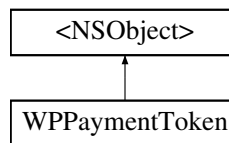
The documentation for this class was generated from the following file:

- /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/Models/WPPaymentInfo.h

5.7 WPPaymentToken Class Reference

```
#import <WPPaymentToken.h>
```

Inheritance diagram for WPPaymentToken:



Instance Methods

- (instancetype) - initWithId:

Properties

- NSString * tokenId

5.7.1 Detailed Description

A [WPPaymentToken](#) represents payment information that was obtained from the user and is stored on [WePay's](#) servers. This token can be used to complete the payment transaction via [WePay's](#) web APIs.

5.7.2 Method Documentation

5.7.2.1 - (instancetype) initWithId: (NSString *) tokenId

Initializes a [WPPaymentToken](#) with the Id provided.

Parameters

<i>tokenId</i>	The Id of the token.
----------------	----------------------

Returns

A [WPPaymentToken](#) object initialized with the Id provided.

5.7.3 Property Documentation**5.7.3.1** `-(NSString*) tokenId` `[read], [nonatomic], [strong]`

The token's id.

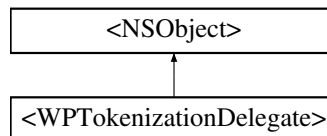
The documentation for this class was generated from the following file:

- /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/Models/WPPaymentToken.h

5.8 <WPTokenizationDelegate> Protocol Reference

```
#import <WePay.h>
```

Inheritance diagram for <WPTokenizationDelegate>:

**Instance Methods**

- (void) - [paymentInfo:didTokenize:](#)
- (void) - [paymentInfo:didFailTokenization:](#)

5.8.1 Detailed Description

This delegate protocol has to be adopted by any class that handles tokenization responses.

5.8.2 Method Documentation**5.8.2.1** `-(void) paymentInfo: (WPPaymentInfo *) paymentInfo didFailTokenization:(NSError *) error`

Called when a tokenization call fails.

Parameters

<i>paymentInfo</i>	The payment info that was provided while making the tokenization call.
--------------------	--

<i>error</i>	The error which caused the failure.
--------------	-------------------------------------

5.8.2.2 - (void) paymentInfo: (WPPaymentInfo *) paymentInfo didTokenize:(WPPaymentToken *) paymentToken

Called when a tokenization call succeeds.

Parameters

<i>paymentInfo</i>	The payment token to be used for completing the transaction.
<i>paymentToken</i>	The payment token representing the payment info.

The documentation for this protocol was generated from the following file:

- /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/WePay.h

6 File Documentation

6.1 /Users/chaitanyabagaria/sandbox/wepay-ios-priv/WePay/WPError.h File Reference

[WPError.h](#) serves as documentation for all errors surfaced by the [WePay](#) iOS SDK.

```
#import <Foundation/Foundation.h>
```

Macros

- `#define WPUnexpectedErrorMessage` `NSStringFromClass(NSLocalizedStringFromTable(@"There was an unexpected error.", @"WePay", @"There was an unexpected error."));`
- `#define WPNoDataReturnedErrorMessage` `NSStringFromClass(NSLocalizedStringFromTable(@"There was no data returned.", @"WePay", @"There was no data returned."));`
- `#define WPCardReaderGeneralErrorMessage` `NSStringFromClass(NSLocalizedStringFromTable(@"Swipe failed due to: (a) uneven swipe speed, (b) fast swipe, (c) slow swipe, or (d) damaged card.", @"WePay", @"Swipe failed due to: (a) uneven swipe speed, (b) fast swipe, (c) slow swipe, or (d) damaged card."));`
- `#define WPCardReaderInitializationErrorMessage` `NSStringFromClass(NSLocalizedStringFromTable(@"Failed to initialize card reader.", @"WePay", @"Failed to initialize card reader."));`
- `#define WPCardReaderTimeoutErrorMessage` `NSStringFromClass(NSLocalizedStringFromTable(@"Card reader timed out.", @"WePay", @"Card reader timed out."));`
- `#define WPSignatureInvalidImageErrorMessage` `NSStringFromClass(NSLocalizedStringFromTable(@"Invalid signature image provided.", @"WePay", @"Invalid signature image provided."));`
- `#define WPNameNotFoundErrorMessage` `NSStringFromClass(NSLocalizedStringFromTable(@"Name not found.", @"WePay", @"Name not found."));`

Enumerations

- `enum WPErrorCode {`
`WPErrorUnknown = -10000, WPErrorNoDataReturned = -10015, WPErrorCardReaderGeneralError = -10016,`
`WPErrorCardReaderInitialization = -10017,`
`WPErrorCardReaderTimeout = -10018, WPErrorCardReaderStatusError = -10019, WPErrorInvalidSignatureImage = -10020,`
`WPErrorNameNotFound = -10021 }`

Variables

- FOUNDATION_EXPORT NSString *const [kWPErrAPIDomain](#)
- FOUNDATION_EXPORT NSString *const [kWPErrSDKDomain](#)
- FOUNDATION_EXPORT NSString *const [kWPErrCategoryKey](#)
- FOUNDATION_EXPORT NSString *const [kWPErrCategoryNone](#)
- FOUNDATION_EXPORT NSString *const [kWPErrCategoryCardReader](#)
- enum [WPErrCode](#) **WPErrCode**

6.1.1 Detailed Description

[WPErr.h](#) serves as documentation for all errors surfaced by the [WePay](#) iOS SDK.

When errors occur, the [WePay](#) iOS SDK returns NSError instances to delegate methods. Each error instance has the following components:

- [error code] gives the integer code corresponding with the error
- [error domain] gives the domain that the error belongs to
- [error userInfo] gives a dictionary with some more useful info, which can be accessed with the keys [kWPErrCategoryKey](#) and [NSLocalizedStringKey](#)

The [WePay](#) iOS SDK can return errors in various error domains:

- [WePay](#) server API errors are in the [kWPErrAPIDomain](#)
- Errors generated by the SDK itself are in the [kWPErrSDKDomain](#)
- System errors generated by iOS are passed through as-is, for example in the [NSURLErrorDomain](#)

See the [WPErrCode](#) section for more details about error codes.

6.1.2 Macro Definition Documentation

6.1.2.1 `#define WPCardReaderGeneralErrorMessage NSLocalizedStringFromTable(@"Swipe failed due to: (a) uneven swipe speed, (b) fast swipe, (c) slow swipe, or (d) damaged card.", @"WePay", @"Swipe failed due to: (a) uneven swipe speed, (b) fast swipe, (c) slow swipe, or (d) damaged card.");`

The localizable user facing message for `WPErrCardReaderGeneralError`, that can be retrieved by calling `[error localizedDescription]`.

6.1.2.2 `#define WPCardReaderInitializationErrorMessage NSLocalizedStringFromTable(@"Failed to initialize card reader.", @"WePay", @"Failed to initialize card reader.");`

The localizable user facing message for `WPErrCardReaderInitialization`, that can be retrieved by calling `[error localizedDescription]`.

6.1.2.3 `#define WPCardReaderTimeoutErrorMessage NSLocalizedStringFromTable(@"Card reader timed out.", @"WePay", @"Card reader timed out.");`

The localizable user facing message for `WPErrCardReaderTimeout`, that can be retrieved by calling `[error localizedDescription]`.


```
6.1.2.4 #define WPNameNotFoundErrorMessage NSLocalizedStringFromTable(@"Name not found.", @"WePay", @"Name not found.");
```

The localizable user facing message for `WPErrorsNameNotFound`, that can be retrieved by calling `[error localizedDescription]`.

```
6.1.2.5 #define WPNoDataReturnedErrorMessage NSLocalizedStringFromTable(@"There was no data returned.", @"WePay", @"There was no data returned.");
```

The localizable user facing message for `WPErrorsNoDataReturned`, that can be retrieved by calling `[error localizedDescription]`.

```
6.1.2.6 #define WPSignatureInvalidImageErrorMessage NSLocalizedStringFromTable(@"Invalid signature image provided.", @"WePay", @"Invalid signature image provided.");
```

The localizable user facing message for `WPErrorsInvalidSignatureImage`, that can be retrieved by calling `[error localizedDescription]`.

```
6.1.2.7 #define WPUnexpectedErrorMessage NSLocalizedStringFromTable(@"There was an unexpected error.", @"WePay", @"There was an unexpected error.");
```

The localizable user facing message for `WPErrorsUnknown`, that can be retrieved by calling `[error localizedDescription]`.

6.1.3 Enumeration Type Documentation

6.1.3.1 enum WPErrorsCode

Error codes for `NSError`s surfaced by the [WePay](#) iOS SDK in the `kWPErrorsSDKDomain`. For a full list of error codes in the `kWPErrorsAPIDomain`, visit <https://www.wepay.com/developer/reference/errors>

Enumerator

`WPErrorsUnknown` -10000 Unknown error.

`WPErrorsNoDataReturned` -10015 No data returned by the API call.

`WPErrorsCardReaderGeneralError` -10016 General error reported by the card reader - usually due to a bad swipe.

`WPErrorsCardReaderInitialization` -10017 Error while initializing the card reader.

`WPErrorsCardReaderTimeout` -10018 Timeout occurred while waiting for card.

`WPErrorsCardReaderStatusError` -10019 Special error reported by card reader - very rare.

`WPErrorsInvalidSignatureImage` -10020 Invalid signature image.

`WPErrorsNameNotFound` -10021 Name not found.

6.1.4 Variable Documentation

6.1.4.1 FOUNDATION_EXPORT NSString* const kWPErrorsAPIDomain

The `NSError` domain of all errors surfaced by the [WePay](#) iOS SDK that were returned by the [WePay](#) API. For a full list of error codes in the `kWPErrorsAPIDomain`, visit <https://www.wepay.com/developer/reference/errors>

6.1.4.2 FOUNDATION_EXPORT NSString* const kWPErrorsCategoryCardReader

The value used in the `NSError`'s `userInfo` dictionary to return the "card reader" error category.

6.1.4.3 FOUNDATION_EXPORT NSString* const kWPErrorCategoryKey

The key used in the NSError's userInfo dictionary to return the error category.

6.1.4.4 FOUNDATION_EXPORT NSString* const kWPErrorCategoryNone

The value used in the NSError's userInfo dictionary to return the "none" error category.

6.1.4.5 FOUNDATION_EXPORT NSString* const kWPErrorSDKDomain

The NSError domain of all errors returned by the [WePay](#) iOS SDK itself. For a full list of error codes in the [kWPError↵ SDKDomain](#), look at [WPErrorCode](#).

Index

WPErr.h

WPErrCardReaderGeneralError, [23](#)

WPErrCardReaderInitialization, [23](#)

WPErrCardReaderStatusError, [23](#)

WPErrCardReaderTimeout, [23](#)

WPErrInvalidSignatureImage, [23](#)

WPErrNameNotFound, [23](#)

WPErrNoDataReturned, [23](#)

WPErrUnknown, [23](#)

WPErrCardReaderGeneralError

WPErr.h, [23](#)

WPErrCardReaderInitialization

WPErr.h, [23](#)

WPErrCardReaderStatusError

WPErr.h, [23](#)

WPErrCardReaderTimeout

WPErr.h, [23](#)

WPErrInvalidSignatureImage

WPErr.h, [23](#)

WPErrNameNotFound

WPErr.h, [23](#)

WPErrNoDataReturned

WPErr.h, [23](#)

WPErrUnknown

WPErr.h, [23](#)