WePay iOS SDK

3.0.0

Generated by Doxygen 1.8.8

Wed Dec 23 2015 01:29:46

Contents

1 Main Page 1

2 Hierarchical Index 7

2.1 Class Hierarchy . . . 7

3 Class Index 7

3.1 Class List . . . 7

4 File Index 8

4.1 File List . . . 8

5 Class Documentation 8

5.1 WePay Class Reference . . . 8

5.1.1 Detailed Description . . . 9

5.1.2 Method Documentation . . . 9

5.1.3 Property Documentation . . . 11

5.2 WPAddress Class Reference . . . 11

5.2.1 Detailed Description . . . 12

5.2.2 Method Documentation . . . 12

5.2.3 Property Documentation . . . 13

5.3 <WPCardReaderDelegate> Protocol Reference . . . 13

5.3.1 Detailed Description . . . 14

5.3.2 Method Documentation . . . 14

5.4 <WPCheckoutDelegate> Protocol Reference . . . 14

5.4.1 Detailed Description . . . 15

5.4.2 Method Documentation . . . 15

5.5 WPConfig Class Reference . . . 15

5.5.1 Detailed Description . . . 16

5.5.2 Method Documentation . . . 16

5.5.3 Property Documentation . . . 17

5.6 WPPaymentInfo Class Reference . . . 17

5.6.1 Detailed Description . . . 18

5.6.2 Method Documentation . . . 18

5.6.3 Property Documentation . . . 19

5.7 WPPaymentToken Class Reference . . . 20

5.7.1 Detailed Description . . . 20

5.7.2 Method Documentation . . . 20

5.7.3 Property Documentation 20
5.8 <WPTokenizationDelegate> Protocol Reference 21
5.8.1 Detailed Description 21
5.8.2 Method Documentation 21

6 File Documentation **21**
6.1 WPError.h File Reference 21
6.1.1 Detailed Description 22
6.1.2 Macro Definition Documentation 23
6.1.3 Enumeration Type Documentation 23
6.1.4 Variable Documentation 24

Index **25**

1 Main Page

Introduction

The WePay 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

Using `cocoapods` (recommended)

- Add `pod "WePay"` to your podfile
- Run `pod install`

- Done!

The SwiftExample app also utilizes cocoapods.

Using library binaries

- Download the latest zip file from our releases page
- Unzip the file and copy the contents anywhere inside your project directory
- 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 framework files you copied:
  - G4XSwiper.framework
  - RPx.framework
  - RUA.framework
  - TrustDefenderMobile.framework
  - WePay.framework
- Done!

Documentation

HTML documentation is hosted on our Github Pages Site.

Pdf documentation is available on the releases page or as a direct download.

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 documentation 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.

See the SwiftExample app for a working implementation of all API methods in a Swift 2 application. Note: make sure to open the project using SwiftApp.xcworkspace and not SwiftApp.xcodeproj.

Initializing a Bridging Header (for Swift apps)

- For using Objective-C modules in a Swift application, you will need to create a bridging header.
- Make sure you are working in {app_name}.xcworkspace file.
- Under your target application folder, create a header file: {app_name}-Bridging-Header.h
- In the Header file, import the modules you need:

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

- Click on the main application project to get to Build Settings.
- Search for bridging header in your target application to find a setting called Swift Compiler - Code Generation.
- Double click in the column next to Objective-C Bridging Header and add your Header file: {app_name}/{app_name}-Bridging-Header.h
- There's no need to import the module in your code; you can use the module by calling it directly in your Swift application.

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 NSLocationWhenInUse↩ UsageDescription.

```
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
```

- Thats 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];
```

- Thats 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];
```

- Thats 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 8

WPAddress 11

<**WPCardReaderDelegate**> 13

<**WPCheckoutDelegate**> 14

WPConfig 15

WPPaymentInfo 17

WPPaymentToken 20

<**WPTokenizationDelegate**> 21

3 Class Index

3.1 Class List

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

WePay 8

WPAddress 11

<**WPCardReaderDelegate**> 13

<**WPCheckoutDelegate**> 14

WPConfig 15

WPPaymentInfo 17

WPPaymentToken 20

<**WPTokenizationDelegate**> 21

4 File Index

4.1 File List

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

WePay.h ??

WPAddress.h ??

WPConfig.h ??

WPError.h
WPError.h serves as documentation for all errors surfaced by the WePay iOS SDK 21

WPPaymentInfo.h ??

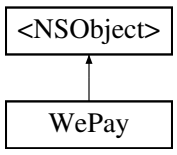
WPPaymentToken.h ??

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 intializer. Use this to initialize the SDK.

Parameters

config	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 timout 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:kWPErrorCategoryCardReader, 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 kWPCard↩ ReaderStatusWaitingForSwipe, 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

cardReader↩ Delegate	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 timout 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:kWPErrorCategoryCardReader, 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 kWPCard↩ ReaderStatusWaitingForSwipe, 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

cardReader↩ Delegate	The delegate class which will receive the response(s) for this call.
tokenization↩ Delegate	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

image	The signature image to be stored.
checkoutId	The checkout id associated with this transaction.
checkout↩ Delegate	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

paymentInfo	The payment info obtained from the user in any form.
tokenization↩ Delegate	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

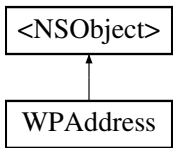
The documentation for this class was generated from the following file:

- WePay.h

5.2 WPAddress Class Reference

`#import <WPAddress.h>`

Inheritance diagram for WPAddress:



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

address1	The first line of the street address.
address2	The second line of the street address.
city	The city.
region	The region. Only for non-US addresses when available.
postcode	The postcode. Only for non-US addresses when available.
country	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

address1	The first line of the street address.
address2	The second line of the street address.
city	The city.
state	The 2-letters US state code.
zip	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

zip	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.

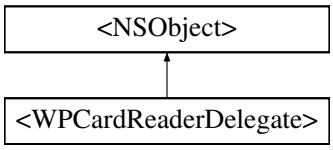
The documentation for this class was generated from the following file:

- 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

status	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

error	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

paymentInfo	The payment info.

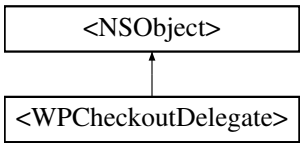
The documentation for this protocol was generated from the following file:

- 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

image	The signature image to be stored.
checkoutId	The checkout id associated with the signature.
error	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

signatureUrl	The url for the signature image.
checkoutId	The checkout id associated with the signature.

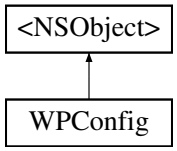
The documentation for this protocol was generated from the following file:

- WePay.h

5.5 WPConfig Class Reference

`#import <WPConfig.h>`

Inheritance diagram for WPConfig:



Instance Methods

- (instancetype) - initWithClientId:environment:
- (instancetype) - initWithClientId:environment:useLocation:restartCardReaderAfterSuccess:restartCardReader↩ AfterGeneralError: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

clientId	Your WePay clientId.
environment	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

clientId	Your WePay clientId.
environment	The environment to be used, one of (kWPEnvironmentStage, kWPEnvironmentProduction).
useLocation	Flag to determine if we should use location services.
restartCard↩ ReaderAfter↩ Success	Flag to determine if the card reader should automatically restart after a successful read.

restartCardReaderAfterGeneralError	Flag to determine if the card reader should automatically restart after a general error (domain:kWPErrorCategoryCardReader, errorCode:WPErrorCardReaderGeneralError).
restartCardReaderAfterOtherErrors	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.

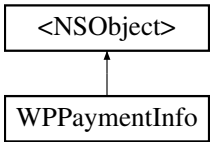
The documentation for this class was generated from the following file:

- 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

email	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

firstName	First name of the payer.
lastName	Last name of the payer.
email	Email address of the payer.
billingAddress	Billing address.

shippingAddress	Shipping address.
cardNumber	The card number.
cvv	The cvv code.
expMonth	The 2-digit expiration month on the credit card.
expYear	The 4-digit expiration year on the credit card.
virtualTerminal	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.

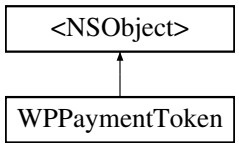
The documentation for this class was generated from the following file:

- 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*

Initialzes a WPPaymentToken with the Id provided.

Parameters

tokenId	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.

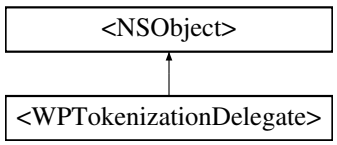
The documentation for this class was generated from the following file:

- 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

paymentInfo	The payment info that was provided while making the tokenization call.
error	The error which caused the failure.

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

Called when a tokenization call succeeds.

Parameters

paymentInfo	The payment token to be used for completing the transaction.
paymentToken	The payment token representing the payment info.

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

- WePay.h

6 File Documentation

6.1 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 NSLocalizedStringFromTable(@"There was an unexpected error.", @"WePay", @"There was an unexpected error.");
- #define WPNoDataReturnedErrorMessage NSLocalizedStringFromTable(@"There was no data returned.", @"WePay", @"There was no data returned.");
- #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.");
- #define WPCardReaderInitializationErrorMessage NSLocalizedStringFromTable(@"Failed to initialize card reader.", @"WePay", @"Failed to initialize card reader.");
- #define WPCardReaderTimeoutErrorMessage NSLocalizedStringFromTable(@"Card reader timed out.", @"WePay", @"Card reader timed out.");
- #define WPSignatureInvalidImageErrorMessage NSLocalizedStringFromTable(@"Inavlid signature image provided.", @"WePay", @"Inavlid signature image provided.");
- #define WPNameNotFoundErrorMessage 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 kWPErrorAPIDomain
- FOUNDATION_EXPORT NSString *const kWPErrorSDKDomain
- FOUNDATION_EXPORT NSString *const kWPErrorCategoryKey
- FOUNDATION_EXPORT NSString *const kWPErrorCategoryNone
- FOUNDATION_EXPORT NSString *const kWPErrorCategoryCardReader
- enum WPErrorCode **WPErrorCode**

6.1.1 Detailed Description

WPError.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 kWPErrorCategoryKey and NSLocalizedDescriptionKey

The WePay iOS SDK can return errors in various error domains:

- WePay server API errors are in the kWPErrorAPIDomain

- Errors generated by the SDK itself are in the kWPErrorSDKDomain
- System errors generated by iOS are passed through as-is, for example in the NSURLErrorDomain

See the WPErrorCode 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 WPErrorCardReaderGeneralError, 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 WPErrorCardReaderInitialization, 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 WPErrorCardReaderTimeout, that can be retrieved by calling [error localized↩ Description].

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

The localizable user facing message for WPErrorNameNotFound, that can be retrieved by calling [error localized↩ Description].

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

The localizable user facing message for WPErrorNoDataReturned, that can be retrieved by calling [error localized↩ Description].

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

The localizable user facing message for WPErrorInavlidSignatureImage, that can be retrieved by calling [error localized↩ Description].

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

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

6.1.3 Enumeration Type Documentation

6.1.3.1 enum WPErrorCode

Error codes for NSErrors surfaced by the WePay iOS SDK in the kWPErrorSDKDomain. For a full list of error codes in the kWPErrorAPIDomain, visit https://www.wepay.com/developer/reference/errors

Enumerator

WPErrorUnknown -10000 Unknown error.

WPErrorNoDataReturned -10015 No data returned by the API call.

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

WPErrorCardReaderInitialization -10017 Error while initializing the card reader.

WPErrorCardReaderTimeout -10018 Timeout occurred while waiting for card.

WPErrorCardReaderStatusError -10019 Special error reported by card reader - very rare.

WPErrorInvalidSignatureImage -10020 Invalid signature image.

WPErrorNameNotFound -10021 Name not found.

6.1.4 Variable Documentation

6.1.4.1 FOUNDATION_EXPORT NSString* const kWPErrorAPIDomain

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 kWPErrorAPIDomain, visit https://www.wepay.com/developer/reference/errors

6.1.4.2 FOUNDATION_EXPORT NSString* const kWPErrorCategoryCardReader

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

WPError.h
    WPErrorCardReaderGeneralError, 24
    WPErrorCardReaderInitialization, 24
    WPErrorCardReaderStatusError, 24
    WPErrorCardReaderTimeout, 24
    WPErrorInvalidSignatureImage, 24
    WPErrorNameNotFound, 24
    WPErrorNoDataReturned, 24
    WPErrorUnknown, 24
WPErrorCardReaderGeneralError
    WPError.h, 24
WPErrorCardReaderInitialization
    WPError.h, 24
WPErrorCardReaderStatusError
    WPError.h, 24
WPErrorCardReaderTimeout
    WPError.h, 24
WPErrorInvalidSignatureImage
    WPError.h, 24
WPErrorNameNotFound
    WPError.h, 24
WPErrorNoDataReturned
    WPError.h, 24
WPErrorUnknown
    WPError.h, 24