Windcave

HIT Module (iOS)

Version 0.4.0

Copyright

Proprietary Notice

Windcave

Document Revision Information and Amendments

All amendments are to be identified and the manual updated, noting the amendment on this amendment page.

Version	Date	Section	Revision Information
0.3.0	10/07/2020	HITWindowStyle	Rename showPxLogo into showLogo
		Methods of HITTransaction class	Add signatureReceiptHandler into startTransaction method
0.4.0	23/07/2020	HITTransactionRecoveryDetails	New class for recovering transaction in progress
		Methods of HITTransaction class	Add recoverTransaction, setDebugMode, setDebugLogHandler
		HITTransactionDetails	Add dpsTxnRef field

Windcave

Contents

1 HIT Module

This module provides methods to control payment transaction on a Windcave terminal. This functionality is known as Host Initiated Transaction (HIT).

The application should invoke methods of static class HITTransaction. This class contains the following methods for starting the transaction, as well as tweaking some additional parameters:

1. startTransaction
   Is used to initiate the transaction

2. setWindowStyle
   This method should be used to tweak the look of the status dialog.

3. getModuleVersion
   This method returns current version of the module

All those methods rely on helper classes instances that you will need to prepare before triggering any of those methods. Those classes are: HITTransactionDetails, HITWindowStyle.

1.1 HITTransactionDetails

The instance of this class should be populated with the initial data for the transaction.

Property name	Value type	Description
amount	NSDecimalNumber	The amount of money that should be withdrawn from user's account.
currencyCode	NSString	Three-character currency code for the transaction (NZD, USD etc.)
deviceId	NSString	HIT Client identifier provided by client. For example, a POS Lane Identifier etc. Alphanumeric, from 1 to 32 characters
stationId	NSString	Station Id unique to the terminal
txnRef	NSString	Set by client to uniquely identify transactions. POS software should persist this value in case the transaction needs to be recovered on POS restart.
merchantRef	NSString	Merchant text field. Max 64 characters (optional)
posName	NSString	Value agreed between POS vendor and Windcave
posVersion	NSString	Version of POS. Supplied by HIT Client to assist transaction recording and diagnosis. Alphanumeric between 1-32 characters
vendorId	NSString	The developer of the POS application. This is agreed between Windcave and vendor. Alphanumeric from 1 to 32 characters in length.
userName	NSString	HIT username provided by Windcave
key	NSString	HIT key provided by Windcave
urlSuccess	NSString	Merchant's back-end URL that should get notifications of successful transactions from Windcave FPRN system

urlFail	NSString	Merchant's back-end URL that should get notifications of failed transactions from Windcave FPRN system
dpsTxnRef	NSString	Windcave transaction reference of the original transaction to perform a matched refund
skipSignatureVerification	BOOL	Specify whether module should automatically accept the signature

All the fields except merchantRef, dpsTxnRef, skipSignatureVerification, urlSuccess and urlFail are mandatory.

1.2 HITTransactionResultDetails

Your code block, that should process result of the transaction, will get the instance of class HITTransactionResultDetails. Here is the list of HITTransactionResultDetails properties that will be accessible for you.

Method name	Description
authCode	Up to 6 characters authorisation code
maskedCardNumber	Masked card number that was used in the transaction
cardName	Card name e.g. Visa
cardNumber2	Windcave generated Luhn-able 16 digit card number alternative id
dpsBillingId	Windcave generated Id for the card
expectedSettlementDate	Expected Settlement Date of Transaction. Format is yyyymmddhhmmss
expectedSettlementDateTimeZone	TimeZone applied to expectedSettlementDate value
transactionDate	Date of Transaction. Returned in Timezone Format is yyyymmddhhmmss
transactionDateTimeZone	TimeZone applied to transactionDate value.
PIX	EMV specific data
RID	EMV specific data
retrievalRefNumber	Retrieval reference number
dpsTxnRef	Windcave transaction reference
responseText	Response text
reCo	Response code
receipt	Receipt that was printed to the customer
receiptLineWidth	Characters per line in the receipt. This is for your reference only, as receipt value is already split on lines
response	Raw response data
transactionResult	Outcome of the transaction (please, see description in the next section)

1.3 HITTransactionResult

Values of this enumeration are used to specify the outcome of the transaction. Here is the current list of values with description:

Value	Description
HITTransactionResultSucceeded	Transaction was approved.

HITTransactionResultFailed	Transaction failed. Some more details could be in the responseText field of HITTransactionResultDetails
HITTransactionResultInvalidTransactionType	Provided transaction type is not supported
HITTransactionResultUserKeyNotMatched	The provided pair of username and key were not found by Windcave
HITTransactionResultStationPrefixError	Station Prefix configured for Station, but not matched with input
HITTransactionResultExistingRequestInProgress	There is another transaction in process
HITTransactionResultTransmitError	Transmission error
HITTransactionResultSessionInitError	Session initialization error. Please check internet connectivity to the pinpad, or try to do a manual logon
HITTransactionResultUserIDNotMatched	HIT User ID not matched
HITTransactionResultTerminalNotConnected	Pinpad is not connected
HITTransactionResultTimeout	Module was unable to get response from the server in time

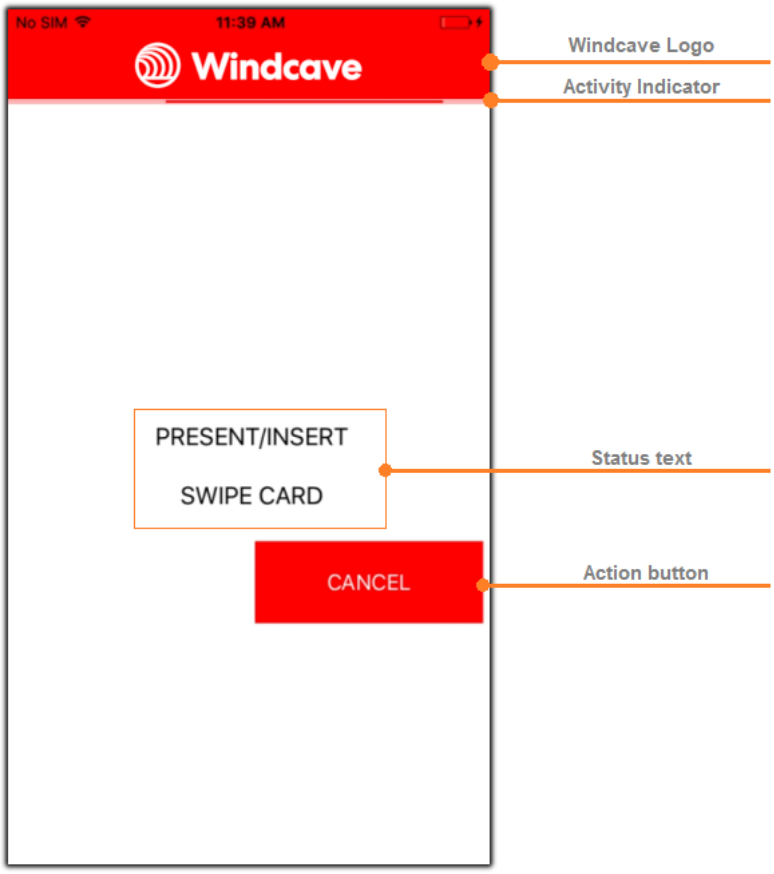
1.4 HITTransactionRecoveryDetails

The instance of this class should be populated with details of the transaction you are attempting to recover. It is useful for cases like POS crash, so that on POS restart you can finish processing of the outstanding transaction.

Property name	Value type	Description
stationId	NSString	Station Id unique to the terminal
txnRef	NSString	The identifier of the transaction that is being recovered
userName	NSString	HIT username provided by Windcave
key	NSString	HIT key provided by Windcave
skipSignatureVerification	BOOL	Specify whether module should automatically accept the signature

1.5 HITWindowStyle

The transaction status dialog has several elements that can be configured depending on the needs of your application. Please keep in mind that on the iPhone the dialog will take the entire screen, while on the iPad it will take only part of screen.



The instance of this class can be used to set up different parameters of the transaction status dialog. The following fields are available for you:

Property name	Description
showLogo	Specify whether you want a Windcave logo at the top of the dialog
showActivityIndicator	Specify whether you want a progress bar to be shown when waiting for response from the server
activityIndicatorColor	The color of activity indicator
backgroundColor	The background color of the dialog
buttonsColor	The background color of action buttons
buttonsTextColor	The text color for the action buttons
buttonsCornerRadius	Corner radius for the action buttons
buttonsFont	The font to be used for action buttons' text
informationTextColor	The color of the status text
informationFont	The font of the status text

1.6 Methods of HITTransaction class

As was mentioned in the beginning of the document, static class HITTransaction should be used to start the transaction. Here are all the methods exposed by this class and some info on their purpose.

```
+ (void) startTransaction: (HITTransactionDetails * _Nonnull)txnDetails
                  txnType: (HITTransactionType)txnType
                parentVC: (UIViewController * _Nonnull)parentVC
                   useUAT: (BOOL)useUAT
            resultHandler: (HITTransactionResultHandler _Nonnull)resultHandler
```

```
  signatureReceiptHandler: (HITTransactionSignatureReceiptHandler
_Nullable)signatureReceiptHandler
```

This method should be triggered to initiate the transaction.

You should pass the following arguments to this method:

`parentVC` – a UIViewController will be used to instantiate the transaction status dialog. This argument cannot be **`nil`**.

`useUAT` – pass YES if you want to perform test transaction on UAT environment.

`txnType` – transaction type can be Purchase, Auth, Complete, Refund or Void

`txnDetails` – transaction details. This argument is also mandatory and cannot be **`null`**.

`resultHandler` – a block of code that will process the result of transaction. This one is also mandatory, cannot be **`null`**.

`signatureReceiptHandler` – a block of code that will process the receipt for cardholder to sign. Can be null.

```
+ (void) recoverTransaction: (HITTransactionRecoveryDetails *
_Nonnull)txnRecoveryDetails
                    parentVC: (UIViewController * _Nonnull)parentVC
                      useUAT: (BOOL)useUAT
               resultHandler: (HITTransactionResultHandler _Nonnull)resultHandler
    signatureReceiptHandler: (HITTransactionSignatureReceiptHandler
_Nullable)signatureReceiptHandler
```

This method should be triggered to recover the status/progress of the outstanding transaction in case POS crashed or has been accidently closed.

You should pass the following arguments to this method:

`parentVC` – a UIViewController will be used to instantiate the transaction status dialog. This argument cannot be **`nil`**.

`useUAT` – pass YES if you want to perform test transaction on UAT environment.

`txnRecoveryDetails` – minimal set of details about the transaction that is being recovered. This argument is also mandatory and cannot be **`null`**.

`resultHandler` – a block of code that will process the result of transaction. This one is also mandatory, cannot be **`null`**.

`signatureReceiptHandler` – a block of code that will process the receipt for cardholder to sign. Can be null.

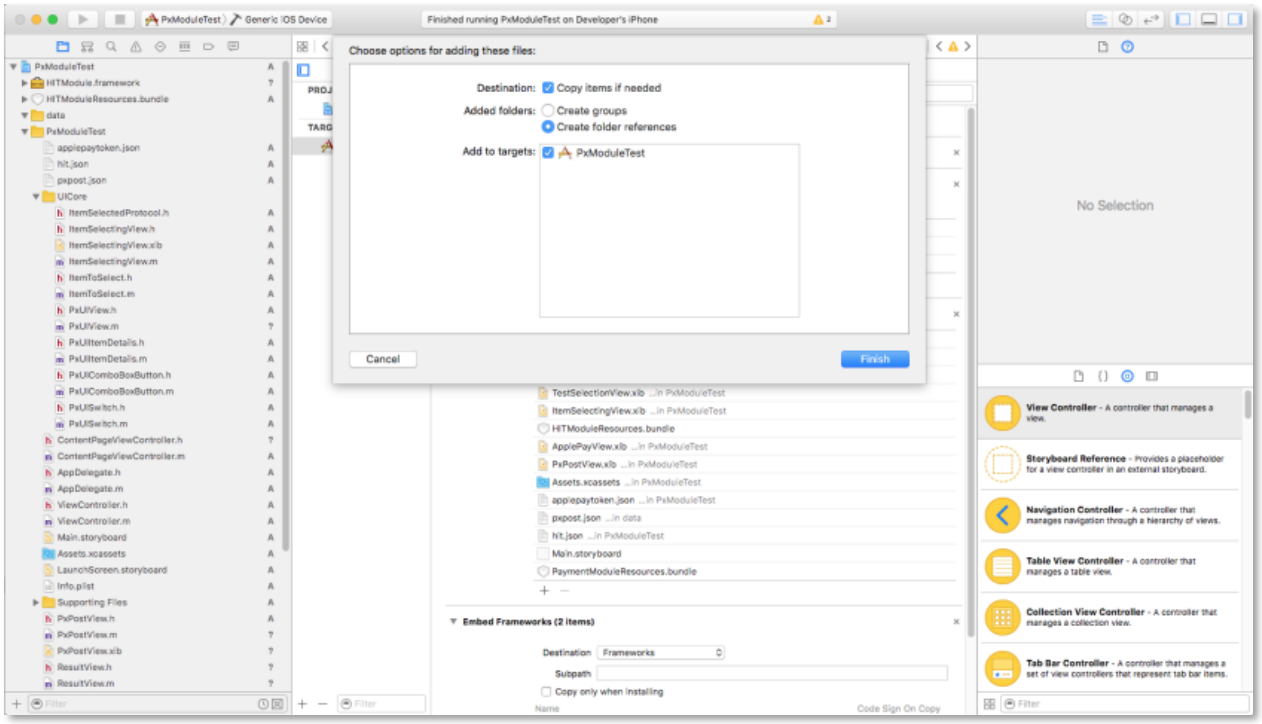
```
+ (void) setWindowStyle: (HITWindowStyle * _Nullable)windowStyle;
```

Invoke this method to tweak the look of the transaction status dialog. If you pass null as `windowStyle` argument, the default settings will be used.

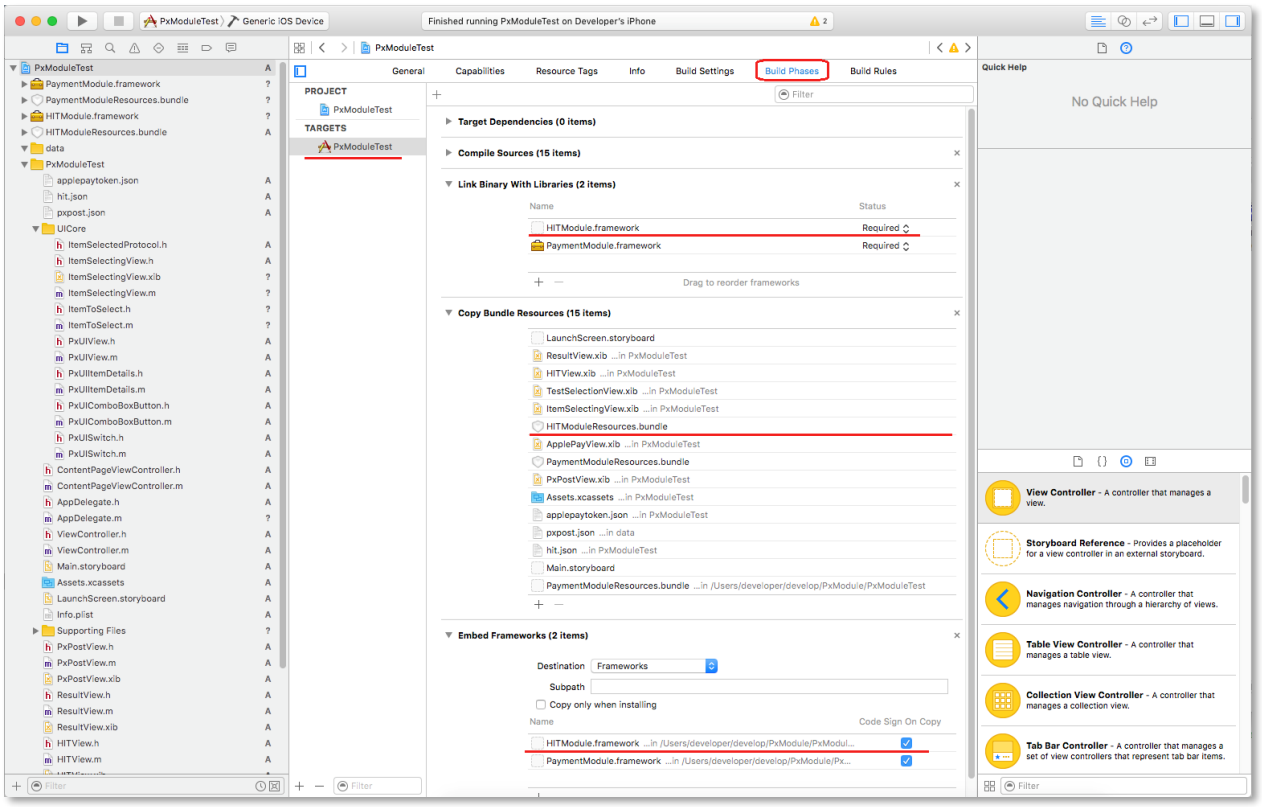
1.7 Integration

Before starting to process transactions, you will need to add **`HITModule.framework`** into your project.

Drag and drop the **`HITModule.framework`** and **`HITModuleResources.bundle`** into your project.



Then click on your project's target, go to **`Build Phases`** tab, and make sure that **`HITModule.framework`** is in **`"Link Binary With Libraries"`** and **`"Embed Frameworks"`**, and that **`HITModuleResources.bundle`** is in **`"Copy Bundle Resources"`**.



After that you can import the header file of the HITModule and start processing transactions:

```
#import "HITModule/HITModule.h"

- (void)startTransaction
{
    HITTransactionDetails *txnDetails = [HITTransactionDetails new];
    txnDetails.userName = @"TestUserName";
    txnDetails.key = @"TestKey";
    txnDetails.amount = [NSDecimalNumber decimalNumberWithString:@"1.00"];
    txnDetails.currencyCode = @"NZD";
    txnDetails.merchantRef = @"MyReference";
    txnDetails.txnRef = @"A00001";
    txnDetails.stationId = @"1234567890";
    txnDetails.deviceId = @"1234567890";
    txnDetails.posName = @"Pos01";
    txnDetails.posVersion = @"Pos v1";
    txnDetails.vendorId = @"TestVendor";

    @try {
        [HITTransaction startTransaction:txnDetails
                                 txnType:HITTransactionTypeAuth
                              parentVC:self
                                 useUAT:YES
                           resultHandler:^(HITTransactionResultDetails *
resultDetails) {
                               // Do result processing here
                           }
                           signatureReceiptHandler:NULL];
    }
    @catch (NSException *exception) {
        // Process exception here
    }
}
```

Important notes

Please keep in mind that if you have got a Universal binary of the framework (the one that work on both, real device and simulator), then you will need to do some before submitting your application to the App Store, as App Store rejects applications that have simulator binaries inside.