Ti.ApplePay

Version 1.1.0 (2016-01-02)

Summary

This module provides access to the native iOS Apply Pay `PassKit` Framework. Using this module, you can easily accept Apple Pay payments using Titanium Mobile. The payment gateway is Stripe (stripe.com), which is recommended by Apple to process payments to your backends easily.

The following features are covered:

- Payment buttons
- Payment request
- Payment dialog
- Payment backend processing using Stripe
- Full compatibility with Titanium Mobile 5.1.0.GA and later

Requirements

- Titanium Mobile SDK 5.0.2.GA or later
- iOS 9 or later

Setup

Unpack the module and place it inside the `/modules/iphone` folder of your project. Edit the modules section of your `tiapp.xml` file to include this module:

```
<modules>
    <module platform="iphone">ti.applepay</module>
</modules>
```

After that, you create an instance of the module by requiring it:

```
var ApplePay = require("ti.applepay");
```

Before you can start using the many different API's, you need to configure the module by providing a Stripe API-Key (from here). The gateways are determined using constants to keep the module open for more payment gateways added in feature releases. The configuration looks like this:

```
ApplePay.setupPaymentGateway({
    name: ApplePay.PAYMENT_GATEWAY_STRIPE,
    apiKey: "<YOUR_STRIPE_API_KEY>"
});
```

You also need the Entitlements.plist with the merchant group's you want to assign. A full tutorial on how to setup the merchant ID's in the developer center can be found here.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/
DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>com.apple.developer.in-app-payments</key>
    <array>
      <string>merchant.de.hansknoechel.paydemo.stripe</string>
      <string>merchant.de.hansknoechel.paydemo.braintree</string>
      <string>merchant.de.hansknoechel.paydemo.chase</string>
    </array>
</dict>
</plist>
```

(Optionally), you can create an 'Apple Pay styled' button by providing the type and style:

```
var payButton = ApplePay.createPaymentButton({
    type: ApplePay.PAYMENT_BUTTON_TYPE_BUY,
    style: ApplePay.PAYMENT_BUTTON_STYLE_WHITE_OUTLINE
});
payButton.addEventListener("click", openPaymentDialog);
```

Now you are able to set properties of the payment request by using the `ApplePay.createPaymentRequest` method:

```
var paymentRequest = ApplePay.createPaymentRequest({
    merchantIdentifier: "merchant.com.company.appname",
    merchantCapabilities: ApplePay.MERCHANT_CAPABILITY_3DS |
ApplePay.MERCHANT_CAPABILITY_CREDIT | ApplePay.MERCHANT_CAPABILITY_DEBIT |
ApplePay.MERCHANT_CAPABILITY_EMV,
    countryCode: "US",
    currencyCode: "USD",
    billingContact: contact,
    shippingContact: contact,
    supportedNetworks: [ApplePay.PAYMENT_NETWORK_VISA,
ApplePay.PAYMENT_NETWORK_MASTERCARD],
    requiredShippingAddressFields: ApplePay.ADDRESS_FIELD_POSTAL_ADDRESS,
    requiredBillingAddressFields: ApplePay.ADDRESS_FIELD_POSTAL_ADDRESS,
    shippingType: ApplePay.SHIPPING_TYPE_DELIVERY,
    shippingMethods: shippingMethods,
    summaryItems: summaryItems,
    applicationData: {
        "userId": 1337
    }
});
```

Although most properties are self-describing, you can find all properties in the official Apple documentation as well, in addition to the detailed code documentation following in the next chapter.

To actually show the payment dialog, you can use the `ApplePay.createPaymentDialog` method:

```
var paymentDialog = ApplePay.createPaymentDialog({
    paymentRequest: paymentRequest
});
paymentDialog.show();
```

There are a couple of payment events you need to implement to respond to changes made in the payment dialog and to finish a payment:

```
paymentDialog.addEventListener("didSelectPayment", didSelectPayment);
paymentDialog.addEventListener("didSelectShippingContact",
didSelectShippingContact);
paymentDialog.addEventListener("didSelectShippingMethod",
didSelectShippingMethod);
paymentDialog.addEventListener("willAuthorizePayment", didAuthorizePayment);
paymentDialog.addEventListener("didAuthorizePayment", willAuthorizePayment);
paymentDialog.addEventListener("close", willClose);

function didSelectPayment(e) {
    e.handler.complete(paymentRequest.getSummaryItems());
}

function didSelectShippingContact(e) {
    e.handler.complete(ApplePay.PAYMENT_AUTHORIZATION_STATUS_SUCCESS,
paymentRequest.getShippingMethods(), paymentRequest.getSummaryItems());
}

function didSelectShippingMethod(e) {
    e.handler.complete(ApplePay.PAYMENT_AUTHORIZATION_STATUS_SUCCESS,
paymentRequest.getSummaryItems());
}

function willAuthorizePayment() {
  // Do amazing stuff here as well
}

function didAuthorizePayment(e) {
    // ... Send the encrypted payment data to your backend
    Ti.API.info("Payment successfully authorized: " + e.success);

    e.handler.complete(ApplePay.PAYMENT_AUTHORIZATION_STATUS_SUCCESS);
}
```

The handler needs to call complete to tell the payment dialog to continue processing. For example, you can change the shipping costs depending on the selected shipping contact or decline a payment if your backend could not be reached.

This makes the module as flexible as it needs to be to handle even advanced payments. An example of using the module in a real-world-application can be found in `example/app.js`.

Documentation

PaymentRequest

A payment request is initialized using the `ApplePay.createPaymentRequest` method.

Properties

- (`String`) merchantIdentifier
  Your merchant identifier, e.g. "`merchant.com.company.app`".

- (`Number`) merchantCapabilities
  A bit field of the payment processing constants (`MERCHANT_CAPABILITY_*`) you support.

- (`String`) countryCode
  The two-letter ISO 3166 country code, e.g. "US".

- (`String`) currencyCode
  The three-letter ISO 4217 currency code, e.g. "USD".

- (`Object`) billingContact
  An object representing a billing contact. Allowed properties:
  - (`String`) firstName
  - (`String`) middleName
  - (`String`) lastName
  - (`String`) prefix
  - (`String`) suffix
  - (`Object`) address
    - (`String`) street
    - (`String`) city
    - (`String`) zip
    - (`String`) state
    - (`String`) country
    - (`String`) ISOCountryCode
  - (`String`) email
  - (`String`) phone

- (`Object`) shippingContact
  An object representing a shipping contact. Allowed properties are same as in billingContact.

- (`Array`) supportedNetworks
  The payment networks that you support, e.g. `[ApplePay.PAYMENT_NETWORK_VISA, ApplePay.PAYMENT_NETWORK_MASTERCARD]`

- (`Number`) requiredShippingAddressFields
  A bit field of shipping address field constants (`ADDRESS_FIELD_*`) that you need in order to process the transaction.
- (`Number`) requiredBillingAddressFields
  A bit field of billing address field constants (`ADDRESS_FIELD_*`) that you need in order to process the transaction.

- (`Number`) shippingType
  The type of shipping used by this request, one of `SHIPPING_TYPE_*`

- (`ShippingMethod`) shippingMethods: shippingMethods
  An array of ApplePay.ShippingMethod objects that describe the supported shipping methods.

- (`[SummaryItem]`) summaryItems: summaryItems
  An array of ApplePay.SummaryItem objects that summarize the amount of the payment.

- (`Object`) applicationData
  Application-specific data or state, e.g. `{"userId": 1337}`

PaymentDialog

A payment dialog is initialized using the `ApplePay.createPaymentDialog` method.

Properties

- (`PaymentRequest`) paymentRequest
  The payment request storing the payment-relevant data.

Methods

- (`void`) open
  Opens the payment dialog modally.

Events

- `didSelectPayment`
  Tells the application that the payment method has changed and asks for a list of updated summary items.

- `didSelectShippingContact`
  Tells the application that the user selected a shipping address.

- `didSelectShippingMethod`
  Tells the application that the user selected a shipping method.

- `willAuthorizePayment`
  Tells the application that the user is authorizing the payment request.

- `didAuthorizePayment`
  Tells the application that the user has authorized the payment request.

- `close`
  Tells the application that payment authorization has completed and the dialog is closed.

PaymentButton

A payment button is initialized using the `ApplePay.createPaymentButton` method.

Properties

- (`Number`) type
  The button's content, one of `PAYMENT_BUTTON_TYPE_*`.

- (`Number`) style
  The button's appearance, one of `PAYMENT_BUTTON_STYLE_*`.

Events

- `click`
  Tells the application that the payment button was clicked.

ShippingMethod

A shipping method is initialized using the `ApplePay.createShippingMethod` method.

Properties

- (`String`) identifier
  A unique identifier for the shipping method, used by the app, e.g. "free_shipping".

- (`String`) title
  A short, localized description of the shipping method, e.g. "Free Shipping".

- (`String`) description
  A user-readable description of the shipping method, e.g. "3-5 working days".

- (`Number`) price
  The shipping method's price, e.g. 99.99

SummaryItem

A summary item is initialized using the `ApplePay.createSummaryItem` method.

Properties

- (`Number`) itemType
  This property defaults to a `PAYMENT_SUMMARY_ITEM_TYPE_FINAL` type. For a list of possible summary item types, see `PAYMENT_SUMMARY_ITEM_TYPE_*`.

- (`String`) title
  A short, localized description of the summary item.

- (`Number`) price
  The summary item's price. **Important**: The total price of the shopping cart is not calculated automatically by the native API. You need to take care of the current amount of the shopping cart. The last summary item of your shopping cart normally is the name of your store and the total amount added together.

Constants

- PAYMENT_BUTTON_TYPE_PLAIN
- PAYMENT_BUTTON_TYPE_BUY
- PAYMENT_BUTTON_TYPE_SETUP

- PAYMENT_BUTTON_STYLE_BLACK
- PAYMENT_BUTTON_STYLE_WHITE
- PAYMENT_BUTTON_STYLE_WHITE_OUTLINE

- PAYMENT_METHOD_TYPE_CREDIT
- PAYMENT_METHOD_TYPE_DEBIT
- PAYMENT_METHOD_TYPE_PREPAID
- PAYMENT_METHOD_TYPE_STORE

- PAYMENT_SUMMARY_ITEM_TYPE_PENDING
- PAYMENT_SUMMARY_ITEM_TYPE_FINAL

- PAYMENT_AUTHORIZATION_STATUS_SUCCESS
- PAYMENT_AUTHORIZATION_STATUS_FAILTURE
- PAYMENT_AUTHORIZATION_STATUS_INVALID_BILLING_POSTAL_ADDRESS
- PAYMENT_AUTHORIZATION_STATUS_INVALID_SHIPPING_POSTAL_ADDRESS
- PAYMENT_AUTHORIZATION_STATUS_INVALID_SHIPPING_CONTACT

- PAYMENT_GATEWAY_NONE
- PAYMENT_GATEWAY_STRIPE
- PAYMENT_GATEWAY_CHASE

- PAYMENT_NETWORK_AMEX
- PAYMENT_NETWORK_DISCOVER
- PAYMENT_NETWORK_MASTERCARD
- PAYMENT_NETWORK_VISA
- PAYMENT_NETWORK_PRIVATE_LABEL
- SHIPPING_TYPE_SHIPPING

- SHIPPING_TYPE_DELIVERY
- SHIPPING_TYPE_SERVICE_PICKUP
- SHIPPING_TYPE_STORE_PICKUP

- ADDRESS_FIELD_NONE
- ADDRESS_FIELD_POSTAL_ADDRESS
- ADDRESS_FIELD_PHONE
- ADDRESS_FIELD_EMAIL
- ADDRESS_FIELD_NAME
- ADDRESS_FIELD_ALL

- MERCHANT_CAPABILITY_3DS
- MERCHANT_CAPABILITY_CREDIT
- MERCHANT_CAPABILITY_DEBIT
- MERCHANT_CAPABILITY_EMV

Further questions?

Feel free to contact us, if you have problems getting the module to run. Happy coding!