iOS Card Management SDK – Network International

Integration Guide Document – Version 1.6

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description of changes |
| 0.1 | 01.10.2022 | Paula Radu | Initial Draft |
| 0.2 | 14.11.2022 | George Cuciureanu | Second draft – UI screenshots added |
| 0.3 | 15.11.2022 | Tudor Stoicovici | Third draft – Quick Start section added |
| 0.4 | 22.11.2022 | George Cuciureanu | 4th Draft – Sections 4.1, 5.5 and 6.1 updated |
| 0.5 | 30.11.2022 | Paula Radu | Updates on section 5.2, 5.3 and 5.4 and 6.  Updated some screenshots. |
| 0.6 | 07.12.2022 | Paula Radu | Updates on image size and updated screenshots; Adjusted GitHub URL |
| 0.7 | 22.12.2022 | Gabriel Cernestean | Updates in sections 5.4.3, 5.4.4 |
| 1.0 | 19.05.2023 | Gabriel Cernestean | Updates in all sections |
| 1.1 | 26.05.2023 | Gabriel Cernestean | Added section 6 |
| 1.2 | 30.05.2023 | Gabriel Cernestean | Updates in section 5 |
| 1.3 | 14.06.2023 | Tudor Stoicovici | Small content arrangements and formatting |
| 1.4 | 14.09.2023 | Gabriel Cernestean | Updates in sections 1, 3, 4, 6 |
| 1.5 | 17.11.2023 | Gabriel Cernestean | Updates as per NI comments |
| 1.6 | 06.12.2023 | Gabriel Cernestean | Added new section 6 |

Contents

[1 Introduction 6](#_Toc152765688)

[1.1 Document Purpose 6](#_Toc152765689)

[1.2 SDK Overview 6](#_Toc152765690)

[1.3 Quick Start 6](#_Toc152765691)

[2 iOS SDK Integration 7](#_Toc152765692)

[2.1 SDK Requirements 7](#_Toc152765693)

[2.2 XCFramework 7](#_Toc152765694)

[2.3 Swift Package Manager 8](#_Toc152765695)

[2.4 CocoaPods 8](#_Toc152765696)

[3 Using the SDK 9](#_Toc152765697)

[3.1 How to Use 9](#_Toc152765698)

[3.2 Expected Input Parameters – NIInput Object 9](#_Toc152765699)

[3.2.1 Bank code 9](#_Toc152765700)

[3.2.2 Card Identifier Id 9](#_Toc152765701)

[3.2.3 Card Identifier Type 9](#_Toc152765702)

[3.2.4 Connection Properties 10](#_Toc152765703)

[3.2.5 Display attributes 10](#_Toc152765704)

[3.2.6 Input Model Example 12](#_Toc152765705)

[3.3 Form Factory Interface 12](#_Toc152765706)

[3.3.1 Display Card Details Form into a Screen 12](#_Toc152765707)

[3.3.2 Display Card Details Form as a View 14](#_Toc152765708)

[3.3.3 Set PIN Form 16](#_Toc152765709)

[3.3.4 Change PIN Form 19](#_Toc152765710)

[3.3.5 Verify PIN Form 22](#_Toc152765711)

[3.3.6 Display View PIN as a view 24](#_Toc152765712)

[3.4 Programatic Interface 26](#_Toc152765713)

[3.4.1 Retrieve Card Details 27](#_Toc152765714)

[3.4.2 Set PIN 27](#_Toc152765715)

[3.4.3 Change PIN 27](#_Toc152765716)

[3.4.4 Verify PIN 28](#_Toc152765717)

[3.4.5 View PIN 28](#_Toc152765718)

[4 SDK User Interface 29](#_Toc152765719)

[4.1 Display Card Details 29](#_Toc152765720)

[4.2 Set PIN Component UI 34](#_Toc152765721)

[4.3 Verify PIN Component UI 36](#_Toc152765722)

[4.4 Change PIN Component UI 36](#_Toc152765723)

[4.5 View PIN Component UI 38](#_Toc152765724)

[4.6 Localization of the SDK (Current Language Support) 38](#_Toc152765725)

[4.7 Light / Dark mode UI 40](#_Toc152765726)

[5 Sample Apps 43](#_Toc152765727)

[6 Appendix – Error handling 44](#_Toc152765728)

[6.1 Server specific errors 44](#_Toc152765729)

[6.1.1 Invalid pin error 44](#_Toc152765730)

[6.2 SDK specific errors 44](#_Toc152765731)

[6.2.1 Navigation error 44](#_Toc152765732)

[6.2.2 Network error 44](#_Toc152765733)

[6.2.3 No data error 44](#_Toc152765734)

[6.2.4 Parsing error 44](#_Toc152765735)

[6.2.5 RSA key error 44](#_Toc152765736)

[6.2.6 Pin encryption error 44](#_Toc152765737)

[7 Appendix – Classes Field Specification (Parameters) 45](#_Toc152765738)

[7.1 NIInput Model 45](#_Toc152765739)

[7.2 NIConnectionProperties 45](#_Toc152765740)

[7.3 NIDisplayAttributes 45](#_Toc152765741)

[7.4 NITheme 45](#_Toc152765742)

[7.5 NILanguage 45](#_Toc152765743)

[7.6 NIFontLabelPair 45](#_Toc152765744)

[7.7 NILabels 46](#_Toc152765745)

[7.8 NICardAttributes 46](#_Toc152765746)

[7.9 NICardDetailsTextPositioning 46](#_Toc152765747)

[7.10 NISuccessResponse 46](#_Toc152765748)

[7.11 NIErrorResponse 46](#_Toc152765749)

# Introduction

## Document Purpose

The purpose of the iOS Software Development Kit (iOS SDK) Integration guide is to serve as a technical documentation for customers that want to integrate with Network International APIs into their mobile applications.

## SDK Overview

The Network International iOS Card Management SDK allows you to quickly display card details, and integrate functionalities like set pin, verify pin, and change pin, into your iOS app.

Network International APIs offer the following operations:

* Display the card details in a view or screen, which contains Clear Card Number (PAN), Expiry Date and Clear CVV2
* Display a PIN-pad to capture and confirm/set the PIN of the card
* Display a PIN-pad to capture the old PIN and new PIN and confirm/change the PIN of the card
* Display a PIN-pad to capture and verify the PIN of the card
* Display the PIN card in a view
* Retrieve card details programmatically
* Set PIN programmatically
* Change PIN programmatically
* Verify PIN programmatically
* View PIN programmatically

## Quick Start

The links below will quickly guide the developer through the most common way of integrating the SDK. For any other option of integration, please check the other sub-chapters as well.

* [**CocoaPods Integration**](#_CocoaPods)
* [**Expected Input Parameters – NIInput O**](#_Expected_Input_Parameters)**bject**
* [**Display Card Details Form as a View**](#_Display_Card_Details)
* [**Set PIN Form**](#_Set_PIN_Form)
* [**Change PIN Form**](#_Change_PIN_Form)
* [**Verify PIN Form**](#_Verify_PIN_Form)
* **View PIN**

# iOS SDK Integration

We support all the popular iOS dependency management tools. The SDK can be added via [CocoaPods](https://cocoapods.org/) or using Swift Package Manager.

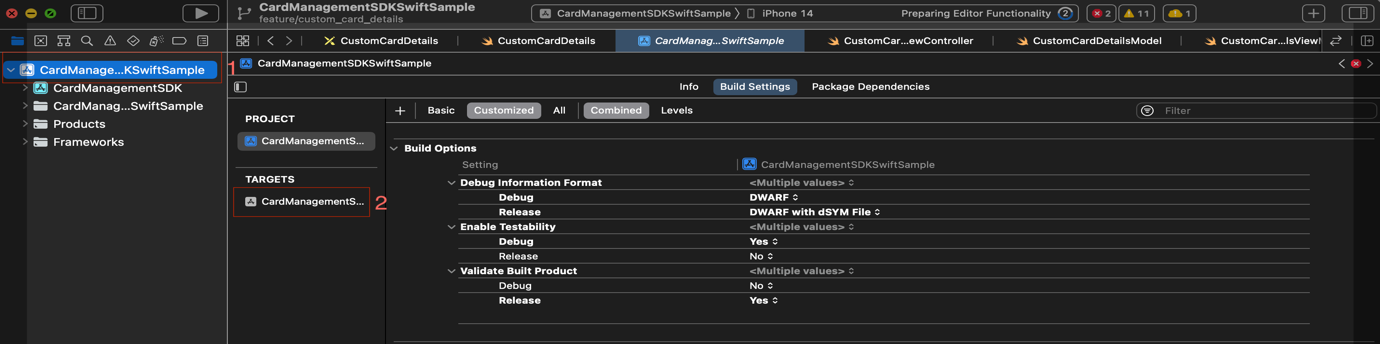
We offer, the possibility to add the XCFramework manually into your project, as well.

## SDK Requirements

* Minimum iOS Version: 12.0
* Xcode: 13 and later
* Swift: 5.x

## XCFramework

Steps to connect our iOS SDK to your iOS application:



1. Navigate to the corresponding project settings by clicking on the project in the project navigator.
2. Select the corresponding target.
3. Drag NICardManagementSDK.xcframework to the *Frameworks, Libraries and Embedded Content* section of your project target: *General ->* *Frameworks, Libraries and Embedded Content*
4. Make sure the “*Embed & Sign*” option is set on Embed tab.

Graphical user interface, application, Teams

Description automatically generated

1. If your application is written with Objective C, perform an additional step: *Build settings -> Build Options ->* *Always Embed Swift Standard Libraries* set YES.

Graphical user interface, text, application

Description automatically generated

## Swift Package Manager

From Xcode simply select *File > Swift Packages > Add Package Dependency...* and paste **https://github.com/network-international/****card-management-sdk-ios** to search field.

## CocoaPods

The steps to install it via CocoaPods:

1. Create a Podfile in the root directory where the .xcodeproj file resides.
2. Add pod 'NICardManagementSDK’ to the Podfile created in the above step. A sample code snippet is shown below. Please note the version below is just for example. Please consult our GitHub repository to find the latest version.

# Uncomment the next line to define a global platform for your project

# platform :ios, '9.0'

target 'NICocoapodsTest' do

# Comment the next line if you don't want to use dynamic frameworks

use\_frameworks!

# Pods for NICocoapodsTest

pod 'NICardManagementSDK', '1.0.18'

end

1. Close Xcode and run pod install command on the directory where the Podfile was created.
2. After performing the above step, open the .xcworkspace file that was created, to open your project.

# Using the SDK

## How to Use

After you have integrated the SDK, by following one of the above set of steps, you can import the SDK into your iOS app and used it.

Swift:

**import** NICardManagementSDK

## Expected Input Parameters – NIInput Object

To use our SDK functionality, the following input is expected:

* Bank Code
* Card Identifier Id
* Card Identifier Type
* Connection Properties
* Display Attributes

### Bank code

* Represents the unique identifier of your financial institution which is initially provided by Network International during onboarding project. This parameter will be passed in all API integration.
* Required parameter

### Card Identifier Id

* Represents the ID of the card
* Required parameter

### Card Identifier Type

* Represents the type of identifier passed.
* Required parameter

Types supported:

* **EXID** (External ID)
* **CONTRACT\_NUMBER** (Clear PAN)

### Connection Properties

* This set of parameters is used to regulate the connectivity between the SDK and your proxy middleware as per the architecture diagram provided by Network International.
* Required parameter
* Connection Properties:
  + rootUrl: Base endpoint of your middleware upon which our URL route will be concatenated.
  + token: Authentication token for your own middleware. This token will be passed in every API call in the header (Authorization) to ensure you perform validation of the identity of the caller.

### Display attributes

* This set of parameters allows for UI customization.
* Display attributes parameter is optional.

Display attributes offered are:

* Theme (dark / light) (theme) - required
* Language (language) - optional
  + Currently supported English and Arabic. If your language is not supported, please contact Network International.
* Fonts (fonts) - optional
* Card Attributes (background image, hide/show property, text positioning) (cardAttributes) - optional

#### Theme

We support dark and light mode, by setting the theme parameter from the display attributes.

If the Customer App is in dark mode, then you should use our SDK with dark theme.

If the Customer App is in light mode, then you should use our SDK with light theme.

#### Language

Languages supported are English and Arabic. You can either set the desired language or not.

The UI will display the words translated in the chosen language.

*E.g.:*

* *If language property is set to English**,* *then the SDK* *forms will be in English*
* *If language property is set to* *Arabic, then the SDK forms will be in Arabic*

If you don’t set any language, it will use the device language, if supported, otherwise will default to English.

*E.g.:*

* *If device language is* *English**, then the SDK forms will be in English*
* *If device language is Arabic, then the SDK forms will be in Arabic*
* *If device language is any other language, then the SDK forms will be in English.*

#### Fonts

We support customization of fonts.

System and custom fonts can be set for the labels of each form view.

#### Card Attributes

Card Attributes is optional. It can be set if customisation of the card details view is wanted.

We offer:

* possibility to show or hide card details by default (required parameter)
* background image customization (optional parameter)
* possibility to set the text position as grouped labels (optional parameter)

For the option to directly show the card details (not masked) when card view is displayed, we expect the shouldHide property to be set to **false**.

For the option to directly hide the card details (masked) when card view is displayed, we expect the shouldHide property to be set to **true**.

If shouldHide property is not set, the default value is **true**.

E.g.:

**let** cardAttributes = NICardAttributes(shouldHide: **false**)

For the card background image, we expect a UIImage to be set, like in the example below:

E.g.:

**let** image = UIImage(named:"background\_image")

**let** cardAttributes = NICardAttributes(shouldHide: **false**, backgroundImage: image))

The image should be added in the Customer App project, in the xcassets catalogue. The recommended size would be 343 x 182.

If all properties are wanted, initialization NICardAttributes is made with all properties.

E.g.:

**let** image = UIImage(named:"background\_image")

**let** textPosition = NICardDetailsTextPositioning(leftAlignment: 0.09, cardNumberGroupTopAlignment: 0.4, dateCvvGroupTopAlignment: 0.6, cardHolderNameGroupTopAlignment: 0.8)

**let** cardAttributes = NICardAttributes(shouldHide: **false**, backgroundImage: image, textPositioning: textPosition)

### Input Model Example

Here’s an example for the input creation with all required and optional parameters.

*Swift*:

**let** rootUrl = [insert your root URL here]

**let** token = [insert your token here]

**let** language = [insert your language here]

**let** theme = [insert your theme here]

**let** image = [insert your UIImage here]

**let** textPosition = [insert your NICardDetailsTextPositioning here]

**let** connectionProperties = NIConnectionProperties(rootUrl: rootUrl, token: token)

**let** fonts = [NIFontLabelPair(font: UIFont(name: "Helvetica", size: 18), label: .setPinDescriptionLabel)]

**let** cardAttributes = NICardAttributes(shouldHide: **false**, backgroundImage: image, textPositioning: textPosition)

**let** displayAttributes = NIDisplayAttributes(theme: theme, language: language, fonts: fonts, cardAttributes: cardAttributes)

**let** input = NIInput(bankCode: bankCode,

cardIdentifierId: cardIdentifierId,

cardIdentifierType: cardIdentifierType,

connectionProperties connectionProperties,

displayAttributes: displayAttributes)

## Form Factory Interface

### Display Card Details Form into a Screen

The form interface will display the card details in a new screen (UIViewController) - navigation style is presentation. The screen can be dismissed by the close button on the top right corner.

The card info displayed are Card Number, Expiry Date, CVV and Cardholder Name.

There is the possibility to copy the card number and card holder name, the option to hide/show card details and also possibility to set card info position on the card view.

#### UI Display

A close-up of a credit card

Description automatically generated A close-up of a credit card

Description automatically generated

#### Function

**func** displayCardDetailsForm(input: NICardManagementSDK.NIInput, viewController: UIViewController, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Parameters:

* **input**: NIInput (see [section 3.2.6](#_Input_Model_Example) for the input model)
* **viewController** is the view controller over which the card details UI should be presented
* **completion**:

- success response

- failure response

- completion: possibility to close form

#### Usage

*Swift:*

NICardManagementAPI.displayCardDetailsForm(input: input, viewController: **self**) { successResponse, errorResponse, closeForm **in**

// Handle error and success

}

### Display Card Details Form as a View

The form interface will display the card details in a standalone view, a custom UIView, named NICardView.

The card info displayed are Card Number, Expiry Date, CVV and Cardholder Name.

There is the possibility to copy the card number and card holder name, the option to hide/show card details and also possibility to set card info position on the card view.

#### UI Display

A close-up of a card

Description automatically generated with medium confidenceA close-up of a credit card

Description automatically generated

#### NICardView class

**class** NICardView : UIView {

/// Initialization of NICardView

/// To be used when creating the card view programmatically

/// - **Parameters**:

/// - input: input needed for the card details visualization

**public** **init**(input: NICardManagementSDK.NIInput, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

/// Set the input for the NICardView

/// To be used ONLY if NICardView is added in storyboard or xib

/// - **Parameters**:

/// - input: input needed for the card details visualization

**public func** setInput(input: NICardManagementSDK.NIInput, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

}

#### Usage - Programmatically

The NICardView can be created programmatically, in code.

Initialization of the NICardView is required with the input parameter, see [section 3.2.6](#_Input_Model_Example) for the input (NIInput) model.

E.g.:

**let** cardView = NICardView(input: cardViewInput) { successResponse, errorResponse, callback **in**

// Handle error and success

}

view.addSubview(cardView)

cardView.translatesAutoresizingMaskIntoConstraints = **false**

Then, the constraints for the NICardView can be added. Below is an example where the card view is placed at the bottom of the screen.

Note that the recommended size is: 345 x 182

NSLayoutConstraint.activate([

cardView.centerXAnchor.constraint(equalTo: view.centerXAnchor),

cardView.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor),

cardView.widthAnchor.constraint(equalToConstant: 345),

cardView.heightAnchor.constraint(equalToConstant: 182),

])

#### Usage – storyboard / xib

The NICardView can be added inside a storyboard or xib.

Add a UIView to your storyboard or xib.

Make sure to set the custom class to **NICardView**, and module to be **NICardManagementSDK**, like so:

Graphical user interface, text, application

Description automatically generated

Connect it in your UIViewController:

**@IBOutlet** **weak** **var** cardView: NICardView!

Then you must set the input, and you’re done!

cardView. setInput(input: input) { successResponse, errorResponse, callback **in**

// Handle error and success

}

Note: Check section 3.2.6 for the input (NIInput) model.

### Set PIN Form

A PIN-pad will be displayed into a separate screen (UIViewController), with ‘push’ navigation style.

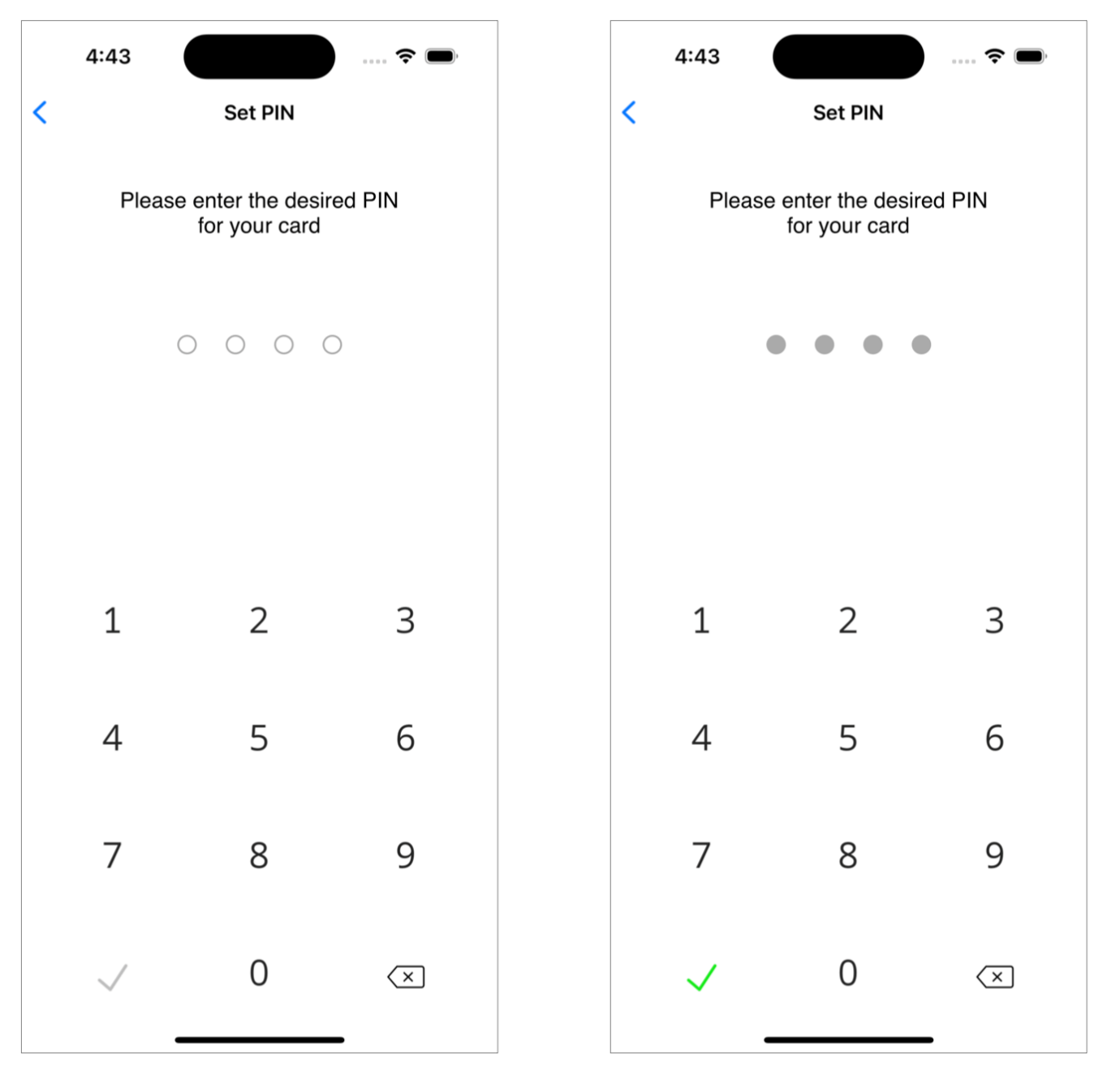
Set PIN is a two-step flow:

1. Capture PIN
2. Confirm PIN

First, the user will need to enter his PIN, pressing the OK (check mark) button from the keyboard, will jump to step 2.

At step 2, the user will need to enter again his PIN for confirmation and press OK (check mark).

#### UI Display



A screenshot of a phone

Description automatically generated with medium confidence A screenshot of a phone

Description automatically generated with low confidence

#### Functions

**func** setPinForm(input: NICardManagementSDK.NIInput, viewController: UIViewController, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

**func** setPinForm(input: NICardManagementSDK.NIInput, type: NICardManagementSDK.NIPinFormType, viewController: UIViewController, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Parameters:

* **input**: NIInput (see [section 3.2.6](#_Input_Model_Example) for the input model)
* **type**: NIPinFormType, is for the PIN length setup: fixed with 4, 5 or 6 digits, or dynamic 4 to 6 digits.
* **viewController** is the view controller from which the PIN-pad screen should navigate from.
* **completion**:

- success response

- failure response

- completion: possibility to close form

#### Usage

*Swift*:

**let** pinType = NIPinFormType.fourDigits // e.g. for pin length of 4 digits

NICardManagementAPI.setPinForm(input: input, type: pinType, viewController: **self**) { successResponse, errorResponse, closeForm **in**

// Handle error and success

}

or without specifying the type (pin length)

NICardManagementAPI.setPinForm(input: input, viewController: **self**) { successResponse, errorResponse, closeForm **in**

// Handle error and success

}

### Change PIN Form

A PIN-pad will be displayed into a separate screen (UIViewController), with ‘push’ navigation style.

Change PIN is a three-step flow:

1. Capture current PIN
2. Capture new PIN
3. Confirm new PIN

First, the user will need to enter his current PIN, pressing the OK (check mark) button from the keyboard, will jump to step 2.

At step 2, the user will need to enter his new PIN and pressing OK (check mark), will jump to step 3.

At step 3, the user needs to enter again the new PIN for confirmation and press OK (check mark).

#### UI Display

Text

Description automatically generated

A picture containing text, screenshot, diagram, design

Description automatically generated

#### Functions

**func** changePinForm(input: NICardManagementSDK.NIInput, viewController: UIViewController, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

**func** changePinForm(input: NICardManagementSDK.NIInput, type: NICardManagementSDK.NIPinFormType, viewController: UIViewController, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Parameters:

* **input**: NIInput (see [section 3.2.6](#_Input_Model_Example) for the input model)
* **type**: NIPinFormType, is for the PIN length setup: fixed with 4, 5 or 6 digits, or dynamic 4 to 6 digits.
* **viewController** is the view controller from which the PIN-pad screen should navigate from.
* **completion**:

- success response

- failure response

- completion: possibility to close form

#### Usage

*Swift:*

**let** pinType = NIPinFormType.sixDigits // e.g. for pin length of 6 digits

NICardManagementAPI.changePinForm(input: input, type: pinType, viewController: **self**) { successResponse, errorResponse, closeForm **in**

// Handle error and success

}

or without specifying the type (pin length)

NICardManagementAPI.changePinForm(input: input, viewController: **self**) { successResponse, errorResponse, closeForm **in**

// Handle error and success

}

### Verify PIN Form

A PIN-pad will be displayed into a separate screen (UIViewController), with ‘push’ navigation style.

The user will need to enter the PIN code, then press the OK (check mark) button from the keyboard.

#### UI Display

A screenshot of a phone

Description automatically generated with medium confidence A screenshot of a phone

Description automatically generated with low confidence

#### Functions

**func** verifyPinForm(input: NICardManagementSDK.NIInput, viewController: UIViewController, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

**func** verifyPinForm(input: NICardManagementSDK.NIInput, type: NICardManagementSDK.NIPinFormType, viewController: UIViewController, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Parameters:

* **input**: NIInput (see [section 3.2.6](#_Input_Model_Example) for the input model)
* **type**: NIPinFormType, is for the PIN length setup: fixed with 4, 5 or 6 digits, or dynamic 4 to 6 digits.
* **viewController** is the view controller from which the PIN-pad screen should navigate from.
* **completion**:

- success response

- failure response

- completion: possibility to close form

#### Usage

*Swift*:

**let** pinType = NIPinFormType.fourDigits // e.g. for pin length of 4 digits

NICardManagementAPI.verifyPinForm(input: input, type: pinType, viewController: **self**) { successResponse, errorResponse, closeForm **in**

// Handle error and success

}

or without specifying the type (pin length)

NICardManagementAPI.verifyPinForm(input: input, viewController: **self**) { successResponse, errorResponse, closeForm **in**

// Handle error and success

}

### Display View PIN as a view

The form interface will display the PIN in a standalone view, a custom UIView, named NIViewPinView.

There is also the possibility to display a timer for which the PIN is visible.

#### UI Display

#### A screenshot of a card Description automatically generated

A screen shot of a card

Description automatically generated

#### NIViewPinView class

**class** NIViewPinView: UIView {

/// Initialization of NIViewPinView

/// To be used when creating the card view programmatically

/// - **Parameters**:

/// - input: input needed for the pin visualization

/// - timer: seconds needed for the pin visualization

**public** **init**(input: NICardManagementSDK.NIInput, timer: Double, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

/// Set the input for the NIViewPinView

/// To be used ONLY if NIViewPinView is added in storyboard or xib

/// - **Parameters**:

/// - input: input needed for the pin visualization

/// - timer: seconds needed for the pin visualization

**public func** setInput(input: NICardManagementSDK.NIInput, timer: Double, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

}

#### Usage - Programmatically

The NIViewPinView can be created programmatically, in code.

Initialization of the NIViewPinView is required with the input parameter and timer parameter, see [section 3.2.6](#_Input_Model_Example) for the input (NIInput) model. For timer parameter is needed a value in seconds:

* 0 for unlimited PIN display time
* Any other value for PIN display time

E.g.:

**let** pinView = NIViewPinView (input: pinViewInput) { successResponse, errorResponse, callback **in**

// Handle error and success

}

view.addSubview(pinView)

#### Usage – storyboard / xib

The NIViewPinView can be added inside a storyboard or xib.

Add a UIView to your storyboard or xib.

Make sure to set the custom class to **NIViewPinView**, and module to be **NICardManagementSDK**, like so:

A screenshot of a phone

Description automatically generated

Connect it in your UIViewController:

**@IBOutlet** **weak** **var** pinView: NIViewPinView!

Then you must set the input, and you’re done!

pinView.setInput(input: input, timer: 5) { successResponse, errorResponse, callback **in**

// Handle error and success

}

Note: Check section 3.2.6 for the input (NIInput) model.

## Programatic Interface

All the above functionality is also offered directly programmatically without using any of the UI components offered by Network International.

**IMPORTANT:**

Please keep in mind that using the programmatic interface will pull you back into the PCI-DSS scope as you will handle sensitive information such as:

* Clear Card Number
* Clear CVV
* Clear PIN

We strongly suggest informing Network International if you are planning to use this integration method.

### Retrieve Card Details

The programmatic interface of the card details will return the card details in an object (NICardDetailsResponse). Will contain card number in clear, expiry date and CVV code in clear, as well as the card holder name.

Function:

**func** getCardDetails(input: NICardManagementSDK.NIInput, completion: **@escaping** (NICardManagementSDK.NICardDetailsResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Success response:

A success message will be sent.

Failure response:

An error code and an error description will be sent.

### Set PIN

The programmatic interface for the Set PIN functionality will return a success or failure response.

The customer application will be responsible to handle the UI part for the PIN capture.

Function:

**func** setPin(pin: String, input: NICardManagementSDK.NIInput, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Success response:

A success message will be sent.

Failure response:

An error code and an error description will be sent.

### Change PIN

The programmatic interface for the Change PIN functionality will return a success or failure response.

The customer application will be responsible to handle the UI part for the PIN capture.

Function:

**func** changePin(oldPin: String, newPin: String, input: NICardManagementSDK.NIInput, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Success response:

A success message will be sent.

Failure response:

An error code and an error description will be sent.

### Verify PIN

The programmatic interface for the Verify PIN functionality will return a success or failure response.

The customer application will be responsible to handle the UI part for the PIN capture.

Function:

**func** verifyPin(pin: String, input: NICardManagementSDK.NIInput, completion: **@escaping** (NICardManagementSDK.NISuccessResponse?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Success response:

A success message will be sent.

Failure response:

An error code and an error description will be sent.

### View PIN

The programmatic interface of the View PIN functionality will return the card PIN or failure response.

Function:

**func** getPin(input: NICardManagementSDK.NIInput, completion: **@escaping** (String?, NICardManagementSDK.NIErrorResponse?, **@escaping** () -> Void) -> Void)

Success response:

A card PIN will be sent.

Failure response:

An error code and an error description will be sent.

# SDK User Interface

## Display Card Details

The Card Details Form API is opening a new UIViewController with a Card Details view in and navigation components on the top.

A close-up of a credit card

Description automatically generated

The Card Details View can be integrated into a screen or another view and represent, actually, the card UI on the screen:

A screenshot of a phone

Description automatically generated with medium confidence

We are exposing some of the scenarios related to parameters for a more clear understanding of how to use NICardAttributes:

* shouldHide = true

A close-up of a credit card

Description automatically generatedA close-up of a credit card

Description automatically generatedA screenshot of a cell phone

Description automatically generated with medium confidence

The *card* *number* and the *card* *holder* *name* have two corresponding buttons with a meaning of “copy to clipboard’ which can be use to copy and share the information.

* shouldHide = false

A close-up of a credit card

Description automatically generated with medium confidence

* backgroundImage

Setting the backgroundImage to a new value will replace the default background image (left) resource of the card with the new one (right):

A close-up of a credit card

Description automatically generated with medium confidence A picture containing text, screenshot, font, design

Description automatically generated

* textPositioning = NICardDetailsTextPositioning(

leftAlignment: 0.09,

cardNumberGroupTopAlignment: 0.4, dateCvvGroupTopAlignment: 0.6,

cardHolderNameGroupTopAlignment: 0.8)

Setting the textPositioning will override the default values (left) with the new one (right):A close-up of a credit card

Description automatically generated with medium confidence A close-up of a credit card

Description automatically generated

All parameters values are interpreted as a percent of the parent container width/height i.e. :

- leftAlignment = 0.09 will be interpreted as 9% of the parent container width

- cardNumberGroupTopAlignment = 0.4 will be interpreted as 40% of the parent container height

## Set PIN Component UI

Set PIN process involves two steps, enter the desired PIN value, and reenter the value for validation process.

If the second value does not match, the user will have to try again until it makes a match with the first value or have the option to navigate back to cancel the process.

A screenshot of a phone

Description automatically generated with medium confidence A screenshot of a phone

Description automatically generated with medium confidence A screenshot of a phone

Description automatically generated with low confidenceA screenshot of a phone

Description automatically generated with low confidence A screenshot of a phone

Description automatically generated with medium confidence A screenshot of a phone

Description automatically generated with low confidence

## Verify PIN Component UI

Verify PIN process is done using a single step. The user enters the value and tap the check mark done button in green.

A screenshot of a phone

Description automatically generated with medium confidence A screenshot of a phone

Description automatically generated with medium confidence A screenshot of a phone

Description automatically generated with low confidence

## Change PIN Component UI

Change PIN is done using one step for the current PIN value and two steps validation for the new PIN value.

If the second value does not match with the first one, the user needs to try until it makes a match, or to navigate back in order to cancel the process.

A screenshot of a phone

Description automatically generated with low confidence A screenshot of a phone

Description automatically generated with low confidence A screenshot of a phone

Description automatically generated with low confidenceA screenshot of a phone

Description automatically generated with low confidence A screenshot of a phone

Description automatically generated with low confidence A picture containing text, screenshot, diagram, design

Description automatically generated

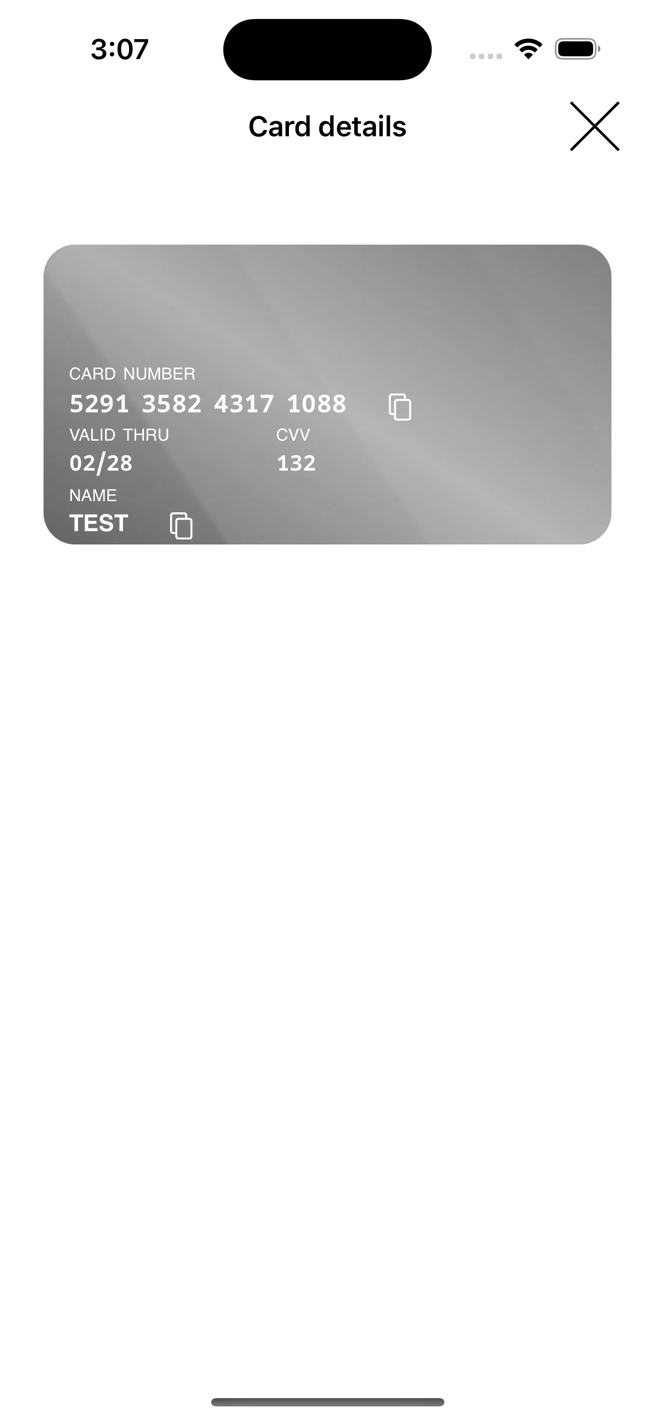
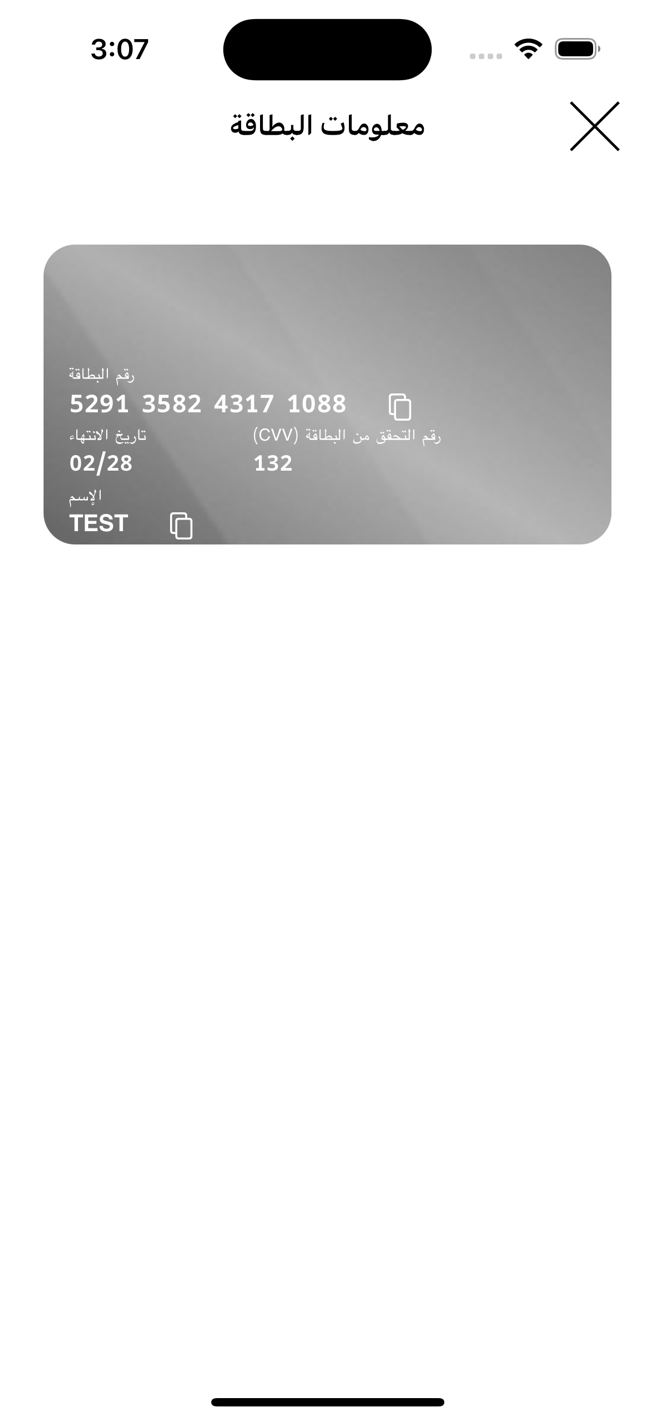
## View PIN Component UI

The View PIN view can be integrated into a screen or another view:

## Localization of the SDK (Current Language Support)

The SDK has support for English (EN) and Arabic (AR) languages, with a default value for English (EN).

Any other options will default to English (EN) locale.

## Light / Dark mode UI

The SDK has support for Light/Dark mode.

A close-up of a credit card

Description automatically generated with medium confidence 

A screenshot of a phone

Description automatically generated with low confidence A screenshot of a cell phone

Description automatically generated

A screenshot of a phone

Description automatically generated with medium confidence A screenshot of a phone

Description automatically generated

A screenshot of a phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

# Sample Apps

There are two sample apps, one written in Swift and the other in Objective-C, available at request, which can be used as a reference for integrating the NICardManagementSDK into your app.

# Appendix – Error handling

## Server specific errors

### Invalid pin error

Occurring when there is an issue with the card PIN.

Error code: 889

Error message: Card/PIN Issue

## SDK specific errors

### Navigation error

Occurring when there is an issue with the navigation towards a screen.

Error message: Form not allowed pushing on navigation controller

### Network error

This is a general error occurring when there an issue with the request to the server or no data is received.

Error message: Server Request Error

### No data error

This is a specific error occurring when the request to the server is successful, but no data is received.

Error message: No Data Found

### Parsing error

Occurring when the data received from the server cannot not be processed.

Error message: SDK Parsing Error

### RSA key error

Occurring when a public key cannot not be generated.

Error message: Couldn't get or generate Public Key

### Pin encryption error

Occurring when there is an issue regarding the pin encryption.

Error message: PIN Block Encryption Error

# Appendix – Classes Field Specification (Parameters)

## NIInput Model

**var** bankCode: String

**var** cardIdentifierId: String

**var** cardIdentifierType: String

**var** connectionProperties: NIConnectionProperties

**var** displayAttributes: NIDisplayAttributes? = **nil**

## NIConnectionProperties

**var** rootUrl: String

**var** token: String

## NIDisplayAttributes

**var** theme: NITheme

**var** language: NILanguage?

**var** fonts: [NIFontLabelPair]?

**var** cardAttributes: NICardAttributes?

## NITheme

**enum** NITheme: Int {

**case** light, dark

}

## NILanguage

**enum** NILanguage: Int {

**case** english

**case** arabic

}

## NIFontLabelPair

**var** font: UIFont

**var** label: NILabels

## NILabels

**enum** NILabels: Int {

/// Card Details

**case** cardNumberLabel

**case** cardNumberValueLabel

**case** expiryDateLabel

**case** expiryDateValueLabel

**case** cvvLabel

**case** cvvValueLabel

**case** cardholderNameLabel

/// Set PIN

**case** setPinDescriptionLabel

/// Change PIN

**case** changePinDescriptionLabel

/// View PIN

**case** viewPinDescriptionLabel

**case** viewPinCountDownDescription

**case** pinDigitLabel

}

## NICardAttributes

**var** shouldHide: Bool? = **true**

**var** backgroundImage: UIImage?

**var** textPositioning: NICardDetailsTextPositioning?

## NICardDetailsTextPositioning

**var** leftAlignment: Double

**var** cardNumberGroupTopAlignment: Double

**var** dateCvvGroupTopAlignment: Double

**var** cardHolderNameGroupTopAlignment: Double

## NISuccessResponse

**var** message: String

## NIErrorResponse

**var** errorCode: String = ""

**var** errorMessage: String = ""

**var** isError = **false**