VICARD iOS SDK

ti&m AG

Version 2.6.3, 2022-03-16

The VICARD SDK allows every Swiss citizen to share data securely with medical persons using their iOS mobile phone.

Overview

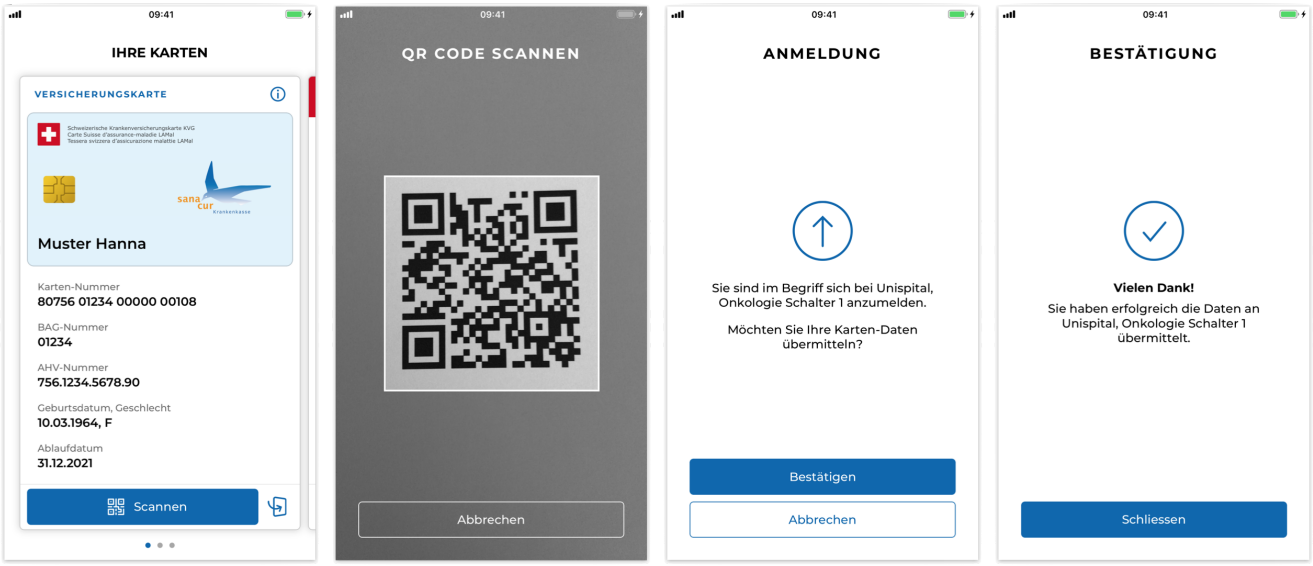
As users are used to access all services using their mobile phone, they expect to be able to do the same within medical processes. Additionally, some digital services cannot be accessed using a smart-card because of technical limitations. This SDK provides a useful alternative for users who do not want to carry a physical card and opens the door to new processes.

Versioning

- Current version of the **VicSdkiOS**: 2.6.3;
- **Minimum iOS version** supported: 9.3;
- **iOS SDK** version: 15.4;

Key features

- *Looks the same* for all insurances, to reduce confusion at medical care providers
- Add several insurance cards, according to the *family structure* of your insurance
- *Check-in* at a medical care provider with the use of a printed QR code
- Receive a strong verification from a *medical person* or an insurance
- Use this verification in other medical processes and web portals
- *Customize* colors and fonts to match your insurance branding
- **In a separate SDK:** allow your users to scan the front or back side of their insurance card



Setup

This SDK supports only native integrations. It's not compatible with any cross-platform solutions. We recommend using CocoaPods to install the VICARD SDK. You can install Cocoapods by following the installation instructions. If you'd rather not use CocoaPods, please refer to your contact person for more integrations possibilities, such as integrating the VICARD SDK frameworks directly.

CocoaPods

1. If you don't have an Xcode project yet, create one now.
2. Create a Podfile if you don't have one:

```
$ cd YOUR-PROJECT-DIRECTORY
$ pod init
```

3. Add the VicSwift pod together with the other pods you want to install. You can include a Pod in your `Podfile` like this:

```
platform :ios, '9.3'

target 'YOUR_APP' do
  use_frameworks!

  pod 'VicSwift', :git =>
'https://<USERNAME>@bitbucket.sasis.ch/scm/vicsdk/vicsdkios.git'
end
```

4. Install the pods and open the `.xcworkspace` file to see the project in Xcode.

```
$ pod install
$ open YOUR-PROJECT.xcworkspace
```

Integration

Permissions

Include the following privacy key to your app's `info.plist` file to allow the usage of the camera:

```
<!-- Required to access mobile's camera -->
<key>NSCameraUsageDescription</key>
<string>YOUR_CUSTOM_REASON_TO_ACCESS_CAMERA</string>
```

Import dependencies

Import the VICARD SDK in your app and customize it:

1. Import the VicSwift module in your `UIApplicationDelegate`:

```
import VicSwift
```

2. Configure the VicSettings and VicCardsSdk shared objects, typically in your application's `application:didFinishLaunchingWithOptions:` method:

```
VicSettings.Typography.FontFamily.regular = "YOUR_CUSTOM_FONT_FAMILY"
VicSettings.Typography.TextStyle.pageTitle.size = YOUR_CUSTOM_FONT_SIZE

let image = UIImage(named: "YOUR_CARD_DESIGN")
VicSettings.Card.setImage(image: image)
```

   For more configuration options, please refer to the Customization section.

Development mode

By default, the VICARD SDK fetches the insurance cards from the production server. In order to test your application against a test server, change the `environment` variable of VicSettings as follows:

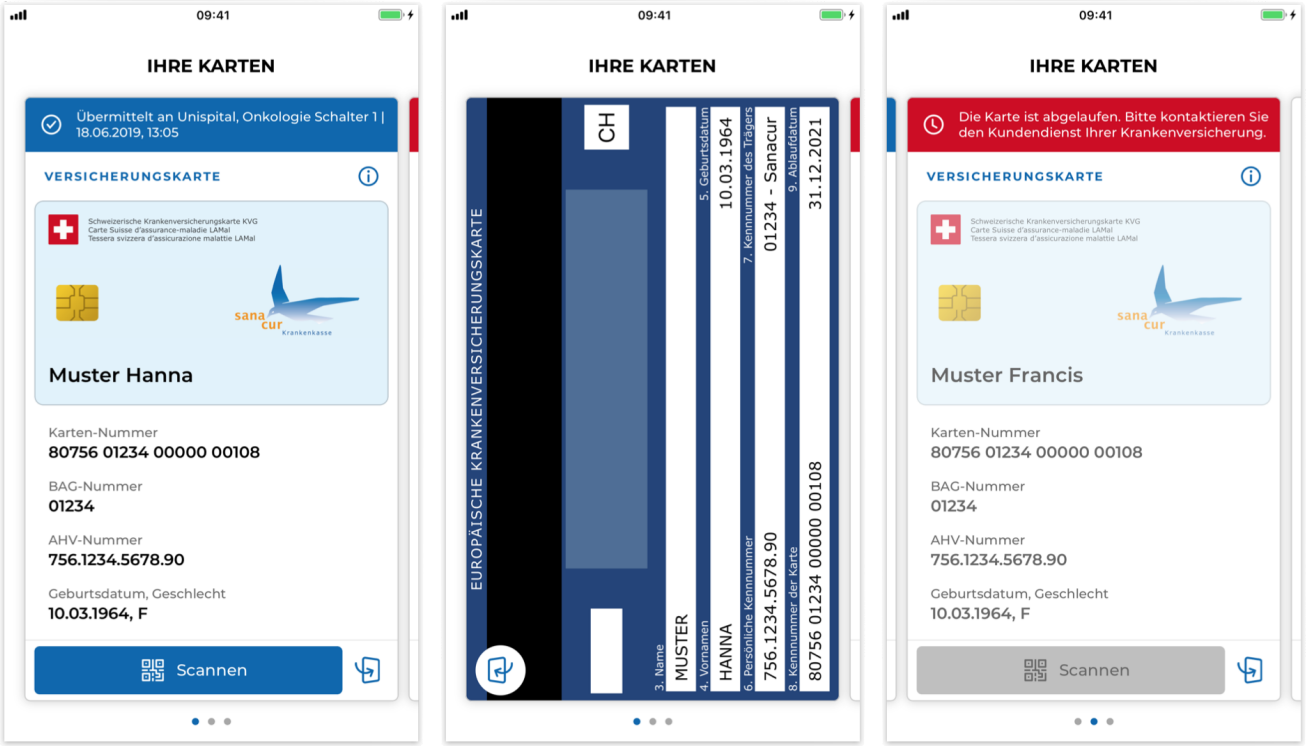
```
// Use mocked server
VicSettings.environment = VicEnv.DEV
```

- The **DEV** environment is a mocked server without any security check. It always returns a list of mocked cards *regardless* of the VekaNr claims set on the SDK.

NOTE Remember to remove the above line when the app is production ready.

Cards

The cards of all the family members can be shown by the VICARD SDK. These can then be used to scan a QR code and transmit a signed/verified version of the card to a third party such as a hospital.



Show cards

Before showing the user's cards, the insurer application needs to fetch the user's **VekaNr claims** from the insurer server and pass them to the VICARD SDK.

NOTE	If you are using the DEV environment, you can simply pass an array containing an empty string to receive a list of mocked cards.

The list of VekaNr claim has to be provided as a JSON array of public PASETOs v2, with one PASETO corresponding to one VekaNr claim. This enables the VICARD SDK to authenticate to the SASIS server and fetch the data of all the cards.

After fetching the VekaNr claims, pass them to the VicCardsSdk class by calling the `setVekaNrClaims` function as follows:

```
VicCardsSdk.setVekaNrClaims(claims: vekaNrClaims)
```

The card view controller can then be shown, for example as a child view controller:

```
override func viewDidLayoutSubviews() {
    if (YOUR_CARD_VIEW.subviews.isEmpty) {
        self.addVicCardsSubview()
    }
}

func addVicCardsSubview() {
    let cardViewController = VicCardViewController(requestUrl: nil)
    addChild(cardViewController)
    cardViewController.view.frame = YOUR_CARD_VIEW.layer.bounds
    YOUR_CARD_VIEW.addSubview(cardViewController.view)
}
```

NOTE YOUR_CARD_VIEW is the view where the cards will be shown.

VICARD errors

The `fetchCards`, `decodeAndValidateCards` and `updateCards` functions return a list of error if something wrong happens during their execution. Here is the list of the possible errors returned by the functions:

Error	Code	Description
Request to get the Cards data failed	100	request failed
Parse cards data failed	101	parsing failed
Validation of the subject field of the card claim failed	102	validation of claim subject failed
Error retrieving the public key of the user	103	get user keypair failed
Creation of the vicard token from the vekaNrClaim failed	104	create token failed
Signing of the vicard token failed	105	sign token failed
Error retrieving the cards data from the local storage	106	get user default key failed

Fetch and validate cards

The VicSdkiOS exposes the functions used to fetch and validate the cards retrieved from the SASIS server.

Use the `VicCardsSdk.fetchCards` function to encode and sign the `vekaNrClaims` set on the SDK and query the SASIS server to retrieve the encoded cards:

```
VicCardsSdk.fetchCards(completionHandler: ([String], [VicError]) -> Void)
```

Use the `VicCardsSdk.decodeAndValidateCards` function to decode the cards returned by the SASIS server and validate them:

```
VicCardsSdk.decodeAndValidateCards(cardsData: [String]) -> ([VicCard], [VicError])
```

Update cards

Use the `VicCardsSdk.updateCards` function to update the cards rendered on the VicCardViewController:

```
// Update cards
VicCardsSdk.updateCards()
```

After the update, all registered `VicCardDelegate`s will be informed of any `VicError` that might have happened during the update.

VicCard Delegate

```
VicCardDelegate {
    //This method will be called whenever the card information is updated
    func cardsUpdateFinished(errors: [VICError])

    //This method will be called on all registered VicCardDelegate whenever a new card
is selected
    func didSelectCardAt(index: Int)

    //This method will be called on all registered VicCardDelegate whenever a new card
is selected
    func didSelectCardWith(number: String)
}
```

Delete cards

As the cards will be saved offline, you need to call the following function to delete all card data when you change from one user to the other or if you allow a user to log out:

```
VicCardsSdk.deleteAll()
```

Card selection

The VicCardViewController exposes the following methods and properties to manually handle the scrolling to specific cards or customize how the card details or tutorial screens are shown:

```
let cardViewController = VicCardViewController(requestUrl: requestUrl, delegate: self)

// Property for enabling/disabling scroll gesture (this also hides page indicators)
cardViewController.isScrollEnabled

// Method to customize the front card details list order or ever to remove items from
the front card view.
// It accepts a list of VicCardListItem in the same order as it is excepted to be
listed inside the card. By omitting one, it will result in hiding the same from the
view.
cardViewController.defineCardListItemOrder([VicCardListItem])

// Method to hide specific tutorial screens. By default the following screens are
displayed: Welcome, QRCode, Barcode, NFC, BackCard.
// It's also possible to hide all tutorial screens by passing to the following method
`VicTutorialScreen.allCases`. In this case, the info button on top of every card will
disappear.
cardViewController.hideTutorialScreen([VicTutorialScreen])

// Property representing the index of the currently displayed insurance card
cardViewController.currentCardIndex
// Method to scroll to a specific card
cardViewController.selectCardAt(index: Int)

// Property representing the number of the currently displayed insurance card
cardViewController.currentCardNumber
// Method to scroll to a specific card
cardViewController.selectCardWith(number: String)
```

User public key

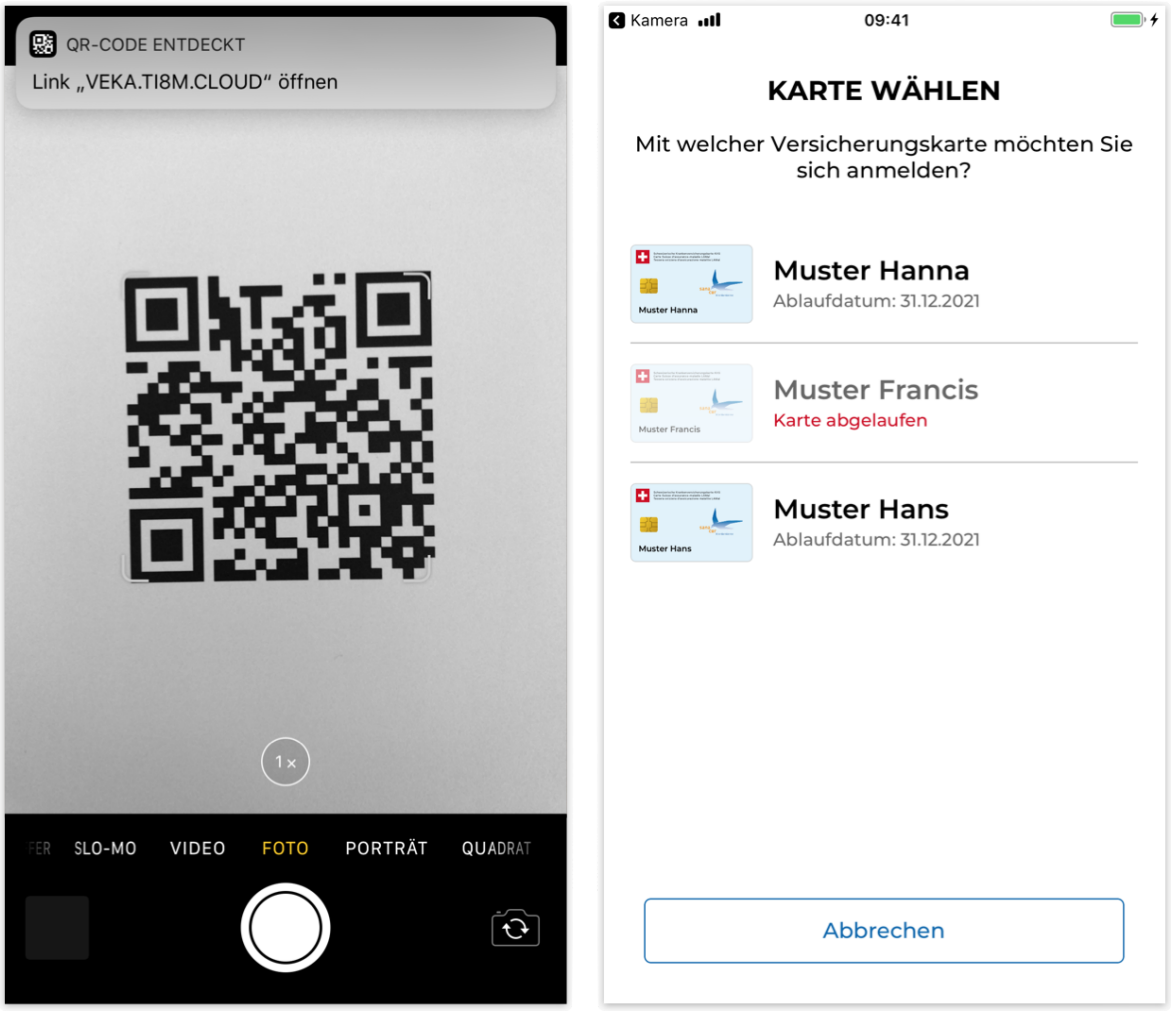
The VICARD SDK securely stores a **public-private keypair** for the user used to sign the VekaNr claims and the selective disclosure response. Call the `getUserPublicKey` function as follows to retrieve the user's public key from the VICARD SDK:

```
let userPublicKey = VicCardsSdk.getUserPublicKey()
```

Universal links / NFC Tags

Universal Links and NFC Tags are useful to speed up the check-in process at a care provider.

- **Universal Links** allow the user to scan the check-in QR code directly from their Camera app;
- **NFC Tags** allow the user to start the check-in process by simply tapping the top of the phone to where the NFC Tag is located (only on iPhone Xs, iPhone Xs Max and iPhone XR).



The following cases need to be handled in order to implement Universal Links and NFC Tags:

1. User is already logged in. The application redirects the user to a card-selection screen where the user will be able to choose which insurance card to check-in.
2. User is not logged in. The application redirects the user to the login screen. After a successful login, the user will be redirected to the card-selection screen.

Implementation

Follow the steps below to handle Universal Links and NFC Tags on your application:

1. Add the Associated Domains Entitlements to your application. The domain you are going to link to your application are related to the QR-Code / NFC Tag the application is going to scan / read. Therefore there are two domains to link, one for the **StageNext** environment and one for the **Prod** environment. If your application has different target per environment, you need to create one .entitlements file for each target and link it to the target on `Targets/[TARGET-NAME]/Build settings/Code signing entitlements`.
    1. Domain used for the StageNext environment: `stagenext.vvk-online.ch`
    2. Domain used for the Prod environment: `www.vvk-online.ch`
2. Send your app ids (in the format of <team identifier>.<bundle identifier>, e.g. `4N2G277B4M.ch.ti8m.vk.sanacur.stagenext`) to your contact person from SASIS. We will add your

app ids to the `apple-app-site-association` file on the StageNext and Prod servers.

3. Handle the data received from the universal link or NFC tag in your `UIApplicationDelegate` as follows:

```
public func application(
    _ application: UIApplication,
    continue userActivity: NSUserActivity,
    restorationHandler: @escaping ([UIUserActivityRestoring]?) -> Void
) -> Bool {
    guard userActivity.activityType == NSUserActivityTypeBrowsingWeb else {
        return false
    }

    if (userActivity.webpageURL != nil) {
        guard let requestUrl = userActivity.webpageURL else {
            return false
        }

        return handleVicLink(requestUrl: requestUrl.absoluteString)
    }

    if #available(iOS 12.0, *) {
        let ndefMessage = userActivity.ndefMessagePayload
        guard
            ndefMessage.records.count > 0,
            ndefMessage.records[0].typeNameFormat != .empty,
            let requestUrl = String(data: ndefMessage.records[0].payload, encoding:
.utf8)
        else {
                return false
        }

        return handleVicLink(requestUrl: requestUrl)
    }

    return true
}

func handleVicLink(requestUrl: String) -> Bool {
    var vc: UIViewController? = nil

    if (CHECK_IF_USER_IS_LOGGED) {
        // User is logged, redirect to the card selection screen
        vc = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "YOUR_CARDS_VIEW_NAME")
        (vc as! YOUR_CARD_VIEW_CONTROLLER).requestUrl = requestUrl
    } else {
        // User is not logged, redirect to the login screen
        vc = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "YOUR_LOGIN_VIEW_NAME")
```

```
        (vc as! YOUR_LOGIN_VIEW_CONTROLLER).requestUrl = requestUrl
    }

    self.window?.rootViewController = vc
    self.window?.makeKeyAndVisible()

    return true
}
```

NOTE Change the above `handleVicLink` function to fit your application.

4. Add the `requestUrl` attribute to the card-list view controller of your application and pass it to the `VicCardViewController`:

```
class YOUR_CARD_VIEW_CONTROLLER: UIViewController {
    [...]
    var requestUrl: String?

    func addVicCardsSubview() {
        let cardViewController = VicCardViewController(requestUrl: requestUrl)
        [...]
    }
}
```

5. Add the `requestUrl` attribute to the login view controller of your application and pass it to the card-list view controller after a successful login:

```
class YOUR_LOGIN_VIEW_CONTROLLER: UIViewController {
    [...]
    var requestUrl: String? = nil

    func onSuccessfulLogin() {
        let sb = UIStoryboard(name: "YOUR_CARD_VIEW_NAME", bundle: nil)
        let vc = sb.instantiateInitialViewController()! as!
YOUR_CARD_VIEW_CONTROLLER
        vc.requestUrl = self.requestUrl
        self.present(vc, animated: true)
    }
```

Customization

Language

In order to change the language of the VICARD SDK, call the `setLanguage` method of the VicSettings:

```
VicSettings.Language.setLanguage(language: VicLang.EN)
```

The languages supported by the VICARD SDK are: German, English, Italian, French.

Typography

Font Families

In order to customize the font families of the VICARD SDK, change the following settings after initializing the SDK:

```
VicSettings.Typography.FontFamily.light
VicSettings.Typography.FontFamily.regular
VicSettings.Typography.FontFamily.medium
VicSettings.Typography.FontFamily.semibold
VicSettings.Typography.FontFamily.bold
```

Text Styles

In order to customize the text styles of the VICARD SDK, change the following settings after initializing the SDK:

```
VicSettings.Typography.TextStyle.pageTitle
VicSettings.Typography.TextStyle.pageTitleInverted
VicSettings.Typography.TextStyle.subhead
VicSettings.Typography.TextStyle.subheadInverted
VicSettings.Typography.TextStyle.copytext
VicSettings.Typography.TextStyle.copytextInverted
VicSettings.Typography.TextStyle.copytextBold
VicSettings.Typography.TextStyle.copytextBoldInverted
```

Each typography object can be customized by the following settings:

```
family: String
size: UInt8
weight: UIFont.Weight
spacing: Float
uppercase: Bool
color: {
    normal: UIColor
    disabled: UIColor
    error: UIColor
}
```

Buttons

In order to customize the buttons of the VICARD SDK, change the following settings after initializing the SDK:

```
VicSettings.Button.primary
VicSettings.Button.primaryInverted
VicSettings.Button.secondary
VicSettings.Button.secondaryInverted
```

Each button object can be customized by the following settings:

```
text: {
    family: String
    size: UInt8
    weight: UIFont.Weight
    spacing: Float
    uppercase: Bool
    color: {
        normal: UIColor
        disabled: UIColor
        selected: UIColor
    }
}
backgroundColor: {
    normal: UIColor
    disabled: UIColor
    selected: UIColor
}
border: {
    width: UInt8,
    radius: UInt8,
    color: {
        normal: UIColor
        disabled: UIColor
        selected: UIColor
    }
}
iconColor: {
    normal: UIColor
    disabled: UIColor
    selected: UIColor
}
```

Icons

In order to customize the icons color of the VICARD SDK, change the following settings after

initializing the SDK:

```
VicSettings.Icon.close
VicSettings.Icon.info
VicSettings.Icon.flip
VicSettings.Icon.error
VicSettings.Icon.system
VicSettings.Icon.systemInverted
```

Each icon object can be customized by the following settings:

```
color: UIColor
```

Cards

In order to customize the cards of the VICARD SDK, change the following settings after initializing the SDK:

```
VicSettings.Card.style
VicSettings.Card.freeField
VicSettings.Card.setImage(image: UIImage, design: VicSettings.Card.Image, cardNumber:
String)
VicSettings.Card.setImageTextColor(color: VicLabelColorState, cardNumber: String)
```

The card style object can be customized by the following settings:

```
front: {
    title: {
        family: String
        size: UInt8
        weight: UIFont.Weight
        spacing: Float
        uppercase: Bool
        color: {
            normal: UIColor
            disabled: UIColor
            error: UIColor
        }
    },
    label: {
        family: String
        size: UInt8
        weight: UIFont.Weight
        spacing: Float
        uppercase: Bool
        color: {
```

```
            normal: UIColor
            disabled: UIColor
            error: UIColor
        }
    },
    text: {
        family: String
        size: UInt8
        weight: UIFont.Weight
        spacing: Float
        uppercase: Bool
        color: {
            normal: UIColor
            disabled: UIColor
            error: UIColor
        }
    }
},
complementaryBack: {
    freeField: {
        family: String
        size: UInt8
        weight: UIFont.Weight
        spacing: Float
        uppercase: Bool
        color: {
            normal: UIColor
            disabled: UIColor
            error: UIColor
        }
    }
},
inner: {
    text: {
        family: String
        size: UInt8
        weight: UIFont.Weight
        spacing: Float
        uppercase: Bool
        color: {
            normal: UIColor
            disabled: UIColor
            error: UIColor
        }
    },
    border: {
        width: UInt8,
        radius: UInt8,
        color: {
            normal: UIColor
            disabled: UIColor
```

```
            error: UIColor
        }
    }
}
banner: {
    text: {
        family: String
        size: UInt8
        weight: UIFont.Weight
        spacing: Float
        uppercase: Bool
        color: {
            normal: UIColor
            disabled: UIColor
            error: UIColor
        }
    },
    backgroundColor: {
        normal: UIColor
        disabled: UIColor
        error: UIColor
    },
    iconColor: {
        normal: UIColor
        disabled: UIColor
        error: UIColor
    }
},
selection: {
    label: {
        family: String
        size: UInt8
        weight: UIFont.Weight
        spacing: Float
        uppercase: Bool
        color: {
            normal: UIColor
            disabled: UIColor
            error: UIColor
        }
    },
    text: {
        family: String
        size: UInt8
        weight: UIFont.Weight
        spacing: Float
        uppercase: Bool
        color: {
            normal: UIColor
            disabled: UIColor
            error: UIColor
```

```
        }
    }
},
backgroundColor: {
    normal: UIColor
    disabled: UIColor
    error: UIColor
},
border: {
    width: UInt8,
    radius: UInt8,
    color: {
        normal: UIColor
        disabled: UIColor
        error: UIColor
    }
}
```

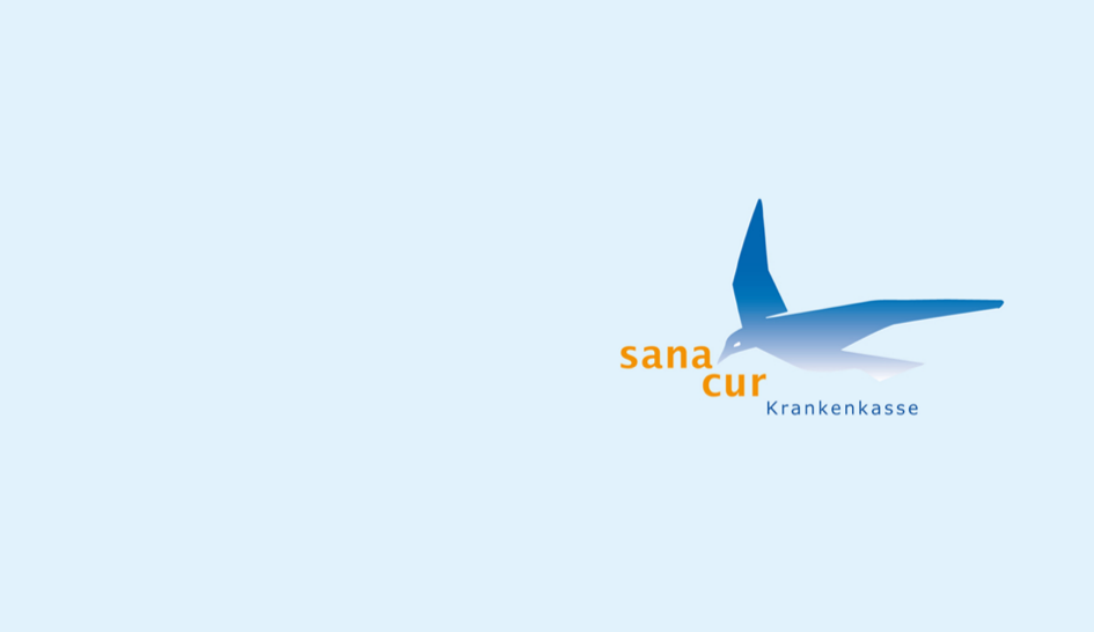
The card freeField attribute holds the text displayed in the free field space of the back side of the cards. To customize it, just set your preferred text like following:

```
VicSettings.Card.freeField.mandatory = "YOUR_MANDATORY_FREE_FIELD_TEXT"
VicSettings.Card.freeField.complementary = "YOUR_COMPLEMENTARY_FREE_FIELD_TEXT"
```

The VICARD SDK comes with one built-in background image, which is used as default for all cards. To customized the card image, you need to add your custom card image to your Xcode assets, such as `Assets.xcassets`. This image must respect following rules:

- it has to be a PNG with the bottom part empty as the card-holder information is put on there dynamically
- it must have a form factor of 1:1.73 e.g.: 945x546 pixel
- the logo must be placed in the right half of the image and must be horizontally right aligned and vertically centered. Its maximum size must be 70% width and 60% height of the total image.

By default, all cards will have the following built-in background image:



You can set the default card background after initializing the SDK:

```
let image = UIImage(named: "YOUR_CARD_DESIGN")
VicSettings.Card.setImage(image: image)
```

To have multiple designs for different cards, a `design` attribute such as `complementaryFront` can be specified:

```
let image = UIImage(named: "YOUR_COMPLEMENTARY_CARD_DESIGN")
VicSettings.Card.setImage(image: image, design: .complementaryFront)
```

It is also possible to set an image for a specific card number with or without specifying a `design` attribute, like so:

```
let defaultImage = UIImage(named: "YOUR_DEFAULT_CARD_DESIGN")
VicSettings.Card.setImage(image: defaultImage, cardNumber: "YOUR_CARD_NUMBER")

let image = UIImage(named: "YOUR_COMPLEMENTARY_CARD_DESIGN")
VicSettings.Card.setImage(image: image, design: .complementaryFront, cardNumber:
"YOUR_CARD_NUMBER")
```

You can even customize inner card text color for a specific card number by calling the following function:

```
---
let simpleTextColor = VicLabelColorState(color: .white)
VicSettings.Card.setImageTextColor(color: simpleTextColor, cardNumber:
"YOUR_CARD_NUMBER")
```

let textColor = VicLabelColorState(normal: .white, disabled: .gray, error: .red) VicSettings.Card.setImageTextColor(color: textColor, cardNumber: "YOUR_CARD_NUMBER") ---

Indicators

In order to customize the indicators of the VICARD SDK, change the following settings after initializing the SDK:

```
VicSettings.Indicator.loading
VicSettings.Indicator.page
VicSettings.Indicator.scroll
```

The loading object can be customized by the following settings:

```
color: UIColor
```

The page object can be customized by the following settings:

```
color: {
    normal: UIColor
    active: UIColor
}
```

The scroll object can be customized by the following settings:

```
color: UIColor
radius: UInt8
```