VIC Android SDK

ti&m AG

v2.6.9 2024-09-03

The VIC SDK allows every Swiss citizen to share data securely with medical persons using their Android 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

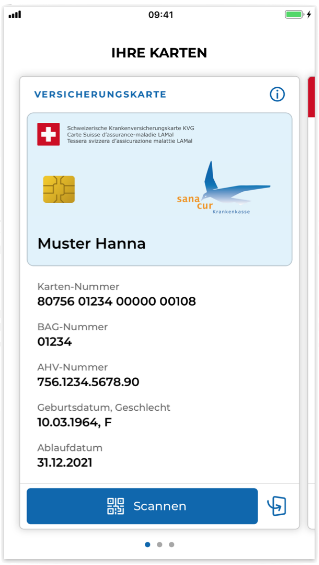
- Current version of the **VicSdkAndroid**: v2.6.9;
- **Minimum Android SDK** version supported: 24;
- Current **Target Android SDK** version: 34;
- Kotlin Version: 1.9.0
- Gradle Version: 8.0

Compatibility

The SDK is tested on multiple different devices from various brands (Samsung, OnePlus, Huawei, Google, ...).

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


09:41
IHRE KARTEN
VERSICHERUNGSKARTE
Schweizerische Krankenversicherungskarte KVG
Carte Suisse d'assurance-maladie LAMal
Tessera svizzera d'assicurazione malattie LAMal
sana cur Krankenkasse
Scannen

Muster Hanna

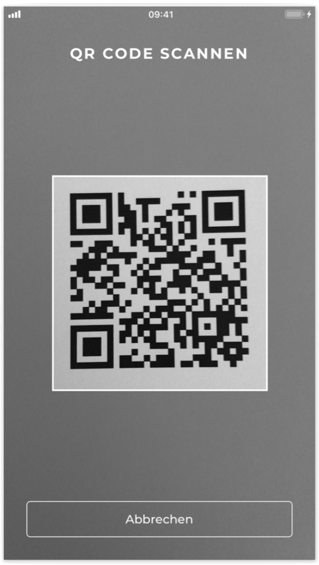
Karten-Nummer
80756 01234 00000 00108

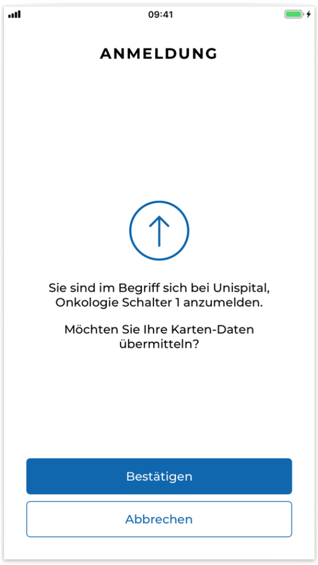
BAG-Nummer
01234

AHV-Nummer
756.1234.5678.90

Geburtsdatum, Geschlecht
10.03.1964, F

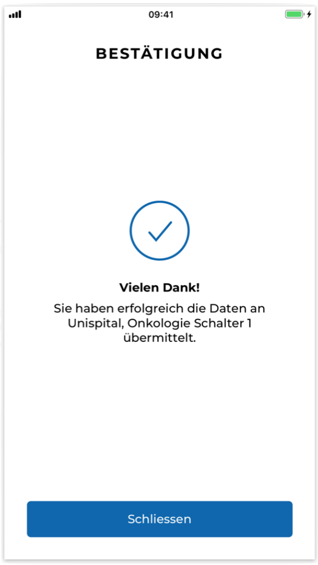
Ablaufdatum
31.12.2021





Sie sind im Begriff sich bei Unispital, Onkologie Schalter 1 anzumelden.

Möchten Sie Ihre Karten-Daten übermitteln?



Vielen Dank!

Sie haben erfolgreich die Daten an Unispital, Onkologie Schalter 1 übermittelt.

Setup

Follow the instruction below to setup the VIC SDK on your Android Studio project.

Download

Download the latest version of the SDK.

```
git clone https://{$USERNAME}@bitbucket.sasis.ch/scm/vicsdk/vicsdkandroid.git
```

Add library

In Android Studio: File → New → New Module → Import AAR Package

select the AAR file of the latest version.

Gradle

1. Include the library module

   settings.gradle

   ```
   include ':app', ':vicsdk'
   ```

2. Include gradle and kotlin-gradle-plugin on root level (Recommended versions in snippet below)

   build.gradle // root-level

   ```
   buildscript {
       dependencies {
           classpath 'com.android.tools.build:gradle:4.2.2'
           classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.9.0"
       }
   ```

```
}
```

3. Include maven repositories on root level

   build.gradle // root-level

```
allprojects {
    repositories {
        google()
        mavenCentral()
    }
}
```

4. insert following packagingOptions, necessary for com.otaliastudios:cameraview:2.7.1 dependency

   build.gradle // app-level

```
android {
...
    packagingOptions {
        exclude 'META-INF/library_release.kotlin_module'
    }
...
}
```

5. For compatibility of Java 8 language APIs with API Levels lower than 26, set coreLibraryDesugaringEnabled to true in compileOptions

   build.gradle // app-level

```
android {
...
    compileOptions {
        coreLibraryDesugaringEnabled true
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
...
}
```

6. Set view binding to true in buildFeatures after compileOptions.

```
android {
...
    buildFeatures {
        viewBinding true
    }
...
}
```

7. Include the library and it’s dependencies in the module

   build.gradle // app-level

```
dependencies {

    // VIC SDK dependencies
    api project(":vicsdk")
    coreLibraryDesugaring 'com.android.tools:desugar_jdk_libs:2.0.3'
    implementation 'com.kazakago.cryptore:cryptore:1.4.0'
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'androidx.recyclerview:recyclerview:1.3.1'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'androidx.preference:preference-ktx:1.2.1'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation "com.goterl:lazysodium-android:5.0.2@aar"
    implementation "net.java.dev.jna:jna:5.8.0@aar"
    implementation "com.google.zxing:core:3.4.0"
    implementation 'commons-codec:commons-codec:1.11'
    implementation 'com.google.android.material:material:1.9.0'
    implementation 'com.google.code.gson:gson:2.9.0'
    implementation 'com.google.mlkit:barcode-scanning:17.2.0'
    implementation 'com.otaliastudios:cameraview:2.7.1'
    implementation 'com.squareup.okhttp3:okhttp:4.9.0'
    implementation 'com.squareup.okhttp3:logging-interceptor:4.9.0'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:adapter-rxjava2:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'io.reactivex.rxjava2:rxandroid:2.0.1'
...
}
```

Manifest

The cardview currently only supports portrait-mode. So the Activity displaying the cards should force portrait mode in the manifest.

```
<activity android:name=".MyCardsActivity"
    android:screenOrientation="portrait"></activity>
```

ProGuard

The following ProGuard rules are required by the SDK to allow minification

```
-keep class com.sun.jna.** { *; }
-keep class ch.ti8m.vk.vicsdk.** { *; }
-keep class paseto.service.VicToken { *; }
-keep class paseto.service.InsurerClaimToken { *; }
-keep class paseto.service.SelectiveDisclosureRequest { *; }
-keep class paseto.service.SelectiveDisclosureResponse { *; }
-keep class paseto.service.SelectiveDisclosureResponseToken { *; }
```

Integration

Import dependencies

Import the VIC SDK in your app and customize it:

1. Import the `ch.ti8m.vk.vicsdk` package in your `MainApplication`:

   ```
   import ch.ti8m.vk.vicsdk.VicSdk
   ```

2. Configure the VicSdk shared objects, typically in your application's `onCreate` method:

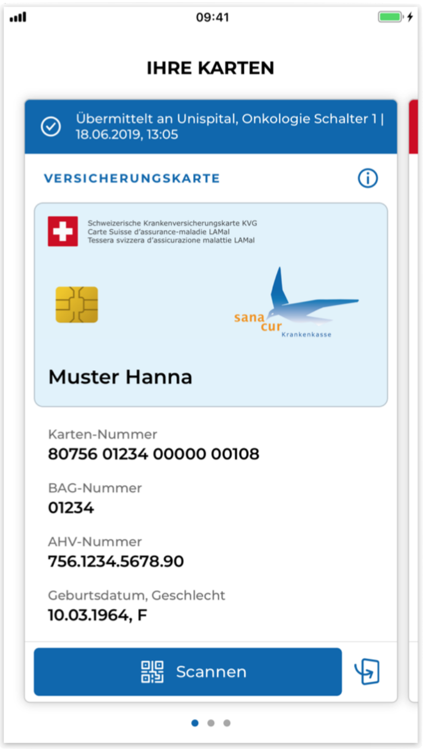
   ```
   override fun onCreate() {
       super.onCreate()

       VicSdk.init(this).setCardImage(your_card_image)
   }
   ```

3. Customize the look and feel of the SDK by overwriting design settings under the `/res/values` folder of your application.

   Please refer to the Customization section.

Cards

The cards of all the family members can be shown by the VIC 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.


09:41

IHRE KARTEN

Übermittelt an Unispital, Onkologie Schalter 1 | 18.06.2019, 13:05

VERSICHERUNGSKARTE

Schweizerische Krankenversicherungskarte KVG
Carte Suisse d'assurance-maladie LAMal
Tessera svizzera d'assicurazione malattie LAMal

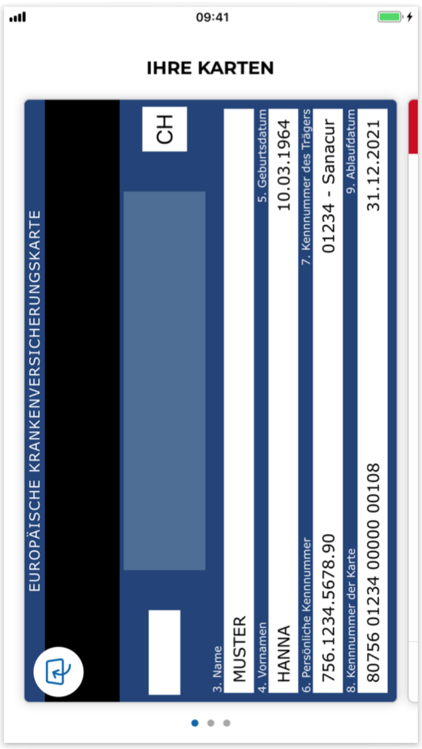
Muster Hanna

Karten-Nummer
80756 01234 00000 00108

BAG-Nummer
01234

AHV-Nummer
756.1234.5678.90

Geburtsdatum, Geschlecht
10.03.1964, F

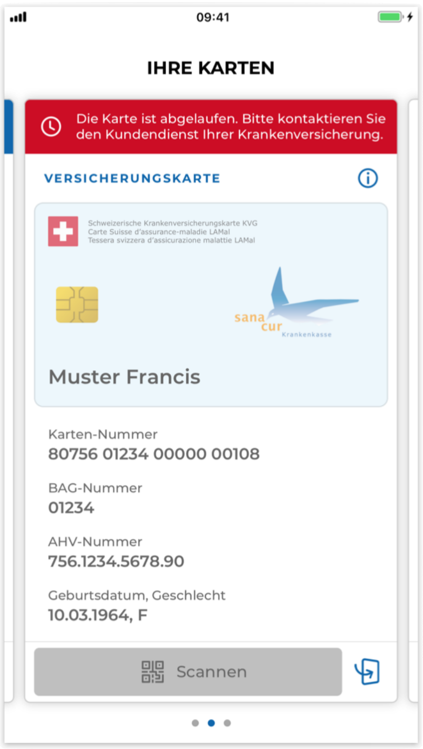

09:41

IHRE KARTEN

EUROPÄISCHE KRANKENVERSICHERUNGSKARTE

CH

3. Name MUSTER
4. Vornamen HANNA
5. Geburtsdatum 10.03.1964
6. Persönliche Kennnummer 756.1234.5678.90
7. Kennnummer des Trägers 01234 - Sanacur
8. Kennnummer der Karte 80756 01234 00000 00108
9. Ablaufdatum 31.12.2021


09:41

IHRE KARTEN

Die Karte ist abgelaufen. Bitte kontaktieren Sie den Kundendienst Ihrer Krankenversicherung.

VERSICHERUNGSKARTE

Schweizerische Krankenversicherungskarte KVG
Carte Suisse d'assurance-maladie LAMal
Tessera svizzera d'assicurazione malattie LAMal

Muster Francis

Karten-Nummer
80756 01234 00000 00108

BAG-Nummer
01234

AHV-Nummer
756.1234.5678.90

Geburtsdatum, Geschlecht
10.03.1964, F

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 VIC 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 VIC SDK to authenticate to the SASIS server and fetch the data of all the cards.

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

```
VicSdk.setVekaNrClaims(vekaNrClaims)
```

The fragment for the cards can then be shown, for example on a FrameLayout element on the card-activity of your application:

```
import ch.ti8m.vk.vicsdk.MyCardsFragment

[...]

val transaction = supportFragmentManager.beginTransaction()
transaction.add(R.id.card_layout, MyCardsFragment())
transaction.commit()
```

```
<FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/card_layout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
```

Vic errors

The `fetchCards`, `updateCards` functions and card getter 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 vic-token from the vekaNrClaim failed	104	create token failed
Signing of the vic-token failed	105	sign token failed
Error retrieving the cards data from the local storage	106	get user default key failed

Card Validation

To update the card information call `VicSdk.fetchCards(context: Context)`. This method is automatically called whenever the MyCardsActivity resumes.

```
// Update cards
VicSdk.fetchCards(context: Context): Observable<Array<Card>>
```

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

Card Listener

```
interface CardListener {
        /** This method will be called on all registered CardListener
            whenever a new card is selected **/
        fun didSelectCardAt(index: Int)
        /** This method will be called whenever the card information is updated **/
        fun cardsUpdateFinished(errors: Array<VicError>)
}
```

Deleting 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:

```
VicSdk.clearCards()
```

Card selection

optional

The VIC SDK exposes the following methods and properties to manually handle scrolling to specific cards.

```
VicSdk.currentCardIndex // this property represents the index of the currently
displayed insurance card

VicSdk.selectCardAt(index: Int) // Method to scroll to a card

VicSdk.addCardListener(listener: CardListener) // Add a CardSelectionListener
VicSdk.removeCardListener(listener: CardListener) // Remove a CardSelectionListener
```

User public key

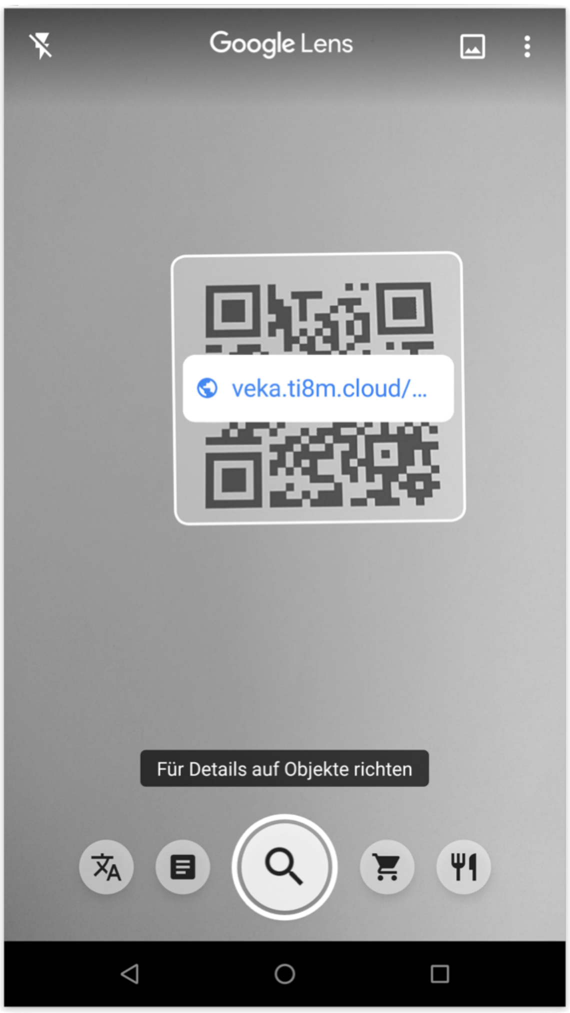
The VIC SDK securely stores a **public-private keypair** for the user used to sign the VekaNr claims and the selective disclosure response. Call the `getUserPublicKey` function as follows to retrieve the user's public key from the VIC SDK:

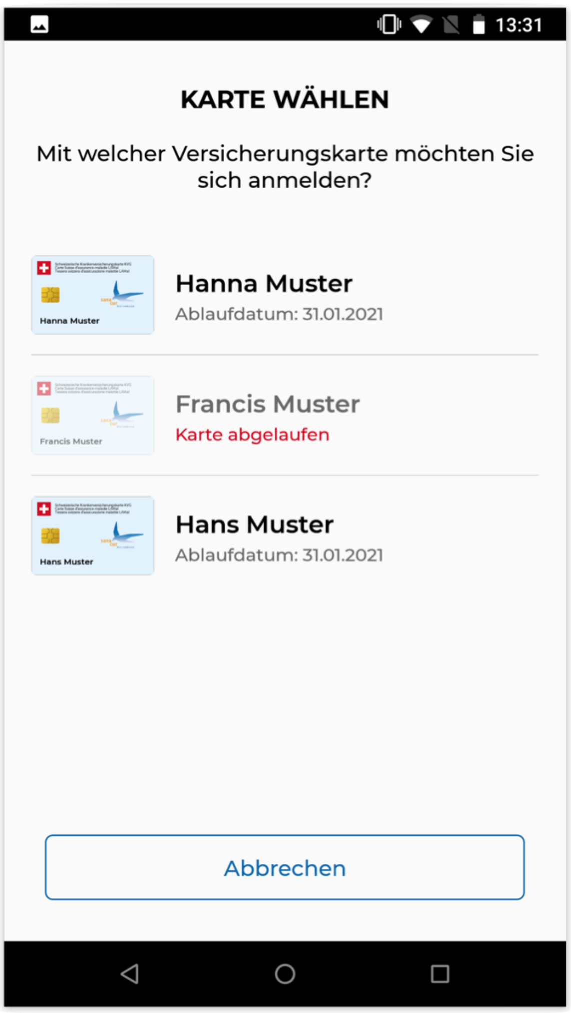
```
val userPublicKey = VicSdk.getUserPublicKey(context)
```

Deep Links / NFC Tags

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

- **Deep Links** allow the user to scan the check-in QR code directly from an application for recognizing QR codes (for example Google Lens);
- **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.




KARTE WÄHLEN

Mit welcher Versicherungskarte möchten Sie sich anmelden?

Hanna Muster
Ablaufdatum: 31.01.2021

Francis Muster
Karte abgelaufen

Hans Muster
Ablaufdatum: 31.01.2021

The following cases need to be handled in order to implement Deep 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 Deep Links and NFC Tags on your application:

1. Add two `intent-filter` tags to your MainActivity class in your AndroidManifest.xml file. The host 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 hosts to link, one for the **StageNext** environment and one for the **Prod** environment. If your application has different flavor per environment, you need set the correct host for the current environment:

   AndroidManifest.xml

```
<activity
    android:name=".MainActivity"
    [...]
>
```

```
    <intent-filter android:autoVerify="true">
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="https" android:host="${appLinkHost}"/>
    </intent-filter>
    <intent-filter android:autoVerify="true">
        <action android:name="android.intent.action.VIEW"/>
        <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="https" android:host="${appLinkHost}"/>
    </intent-filter>
</activity>
```

2. Update your `build.gradle` file in order to set the correct `appLinkHost` for the current environment:

   build.gradle

```
productFlavors {
        [YOUR-STAGE-NEXT-FLAVOR] {
            [...]
            manifestPlaceholders = [
                    [...]
                    appLinkHost: "stagenext.vvk-online.ch"
            ]
        }
        [YOUR-PROD-FLAVOR] {
            [...]
            manifestPlaceholders = [
                    [...]
                    appLinkHost: "www.vvk-online.ch"
            ]
        }
    }
```

3. Send your Application Ids (e.g. `ch.ti8m.vk.sanacur.stageNext`) and your sha256_cert_fingerprints (e.g. `DA:E4:CF:0A:D0:4E:B2:6C:35:DF:7A:DA:2B:CC:AE:B4:9D:71:BA:F8:D4:7A:56:A3:7F:22:BE:8D:84:D4:C6:C7`) to your contact person from SASIS. We will add your app ids to the `assetlinks.json` file on the StageNext and Prod servers.

4. Handle the data received from the deep link or NFC tag in your `MainActivity` class as follows:

```
override fun onCreate(savedInstanceState: Bundle?) {
    if (intent?.data != null) {
        this.handleVicLink()
    }
    [...]
}

private fun handleVicLink() {
    val requestUrl: Uri? = intent?.data
    val intent = if (CHECK_IF_USER_IS_LOGGED) {
        // User is logged, redirect to the card selection screen
        Intent(this@MainActivity, YOUR_CARD_LIST_ACTIVITY::class.java)
    } else {
        // User is not logged, redirect to the login screen
        Intent(this@MainActivity, YOUR_LOGIN_ACTIVITY::class.java)
    }

    intent.putExtra(MyCardsFragment.REQUEST_URL, requestUrl.toString())
    startActivity(intent)
}
```

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

5. Edit the card-list activity of your application to pass the `requestUrl` parameter to MyCardsFragment:

```
class YOUR_CARD_LIST_ACTIVITY: AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        [...]
        val transaction = supportFragmentManager.beginTransaction()
        val bundle = Bundle()
        bundle.putString(MyCardsFragment.REQUEST_URL, intent.
getStringExtra(MyCardsFragment.REQUEST_URL))
        val fragment = MyCardsFragment()
        fragment.arguments = bundle
        transaction.add(R.id.card_layout, fragment)
        transaction.commit()
    }
}
```

6. Edit the login activity of your application to pass the `requestUrl` parameter to your card-list activity after a successful login:

```
class YOUR_LOGIN_ACTIVITY : AppCompatActivity() {
    [...]

    private fun onSuccessfulLogin() {
        val cardsIntent = Intent(this@YOUR_LOGIN_ACTIVITY,
YOUR_CARDS_LIST_ACTIVITY::class.java)
        cardsIntent.flags = Intent.FLAG_ACTIVITY_NEW_TASK
        cardsIntent.putExtra(MyCardsFragment.REQUEST_URL, intent.
getStringExtra(MyCardsFragment.REQUEST_URL))
```

```
        startActivity(cardsIntent)
        this@YOUR_LOGIN_ACTIVITY.finish()
    }
```

Customization

Language

In order to change the language of the VIC SDK, call the `setLanguage` method of the VicSdk class:

```
fun setLanguage(context: Context, language: VicLang, shouldRecreate: Boolean = false)


VicSdk.setLanguage(context, VicLang.DE)
```

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

The Context should be an Activity Context.

The `shouldRecreate` flag determines whether or not the activity (context) should be recreated immediately for the language settings to take immediate effect.

— Attention — If the language of the SDK changes **after** the CardsActivity is created, the Activity must be recreated for the language change to take effect.

To pass the Language-Setting to the Base-Context of the Application, The method `attachBaseContext` has to be overriden in the CardsActivity with the LocaleHelper of the SDK.

```
    override fun attachBaseContext(newBase: Context?) {
        super.attachBaseContext(LocaleHelper.onAttach(newBase!!))
    }
```

Typography

For customizing a specific text, create a resource file under the `/res/values` folder of your application. Then copy-paste the style element of that text type into it and overwrite the specific values.

Make sure to always copy all items of a style element, otherwise it will overwrite items, which were not copied, with default values. Further explanation about the style items follows after the Textview Styles.

Textview Styles

Following Texts can be customized by Font Family, Style, Size, Spacing and Color:

- Page Title

```
<style name="sdk_page_title">
    <item name="android:fontFamily">sans-serif-medium</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textSize">18sp</item>
    <item name="android:textAllCaps">true</item>
    <item name="android:letterSpacing">0.11</item>
    <item name="android:textColor">@color/sdk_text_value_color</item>
</style>
```

- Page Title Inverted

```
<style name="sdk_inverted_page_title">
    <item name="android:fontFamily">sans-serif-medium</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textSize">18sp</item>
    <item name="android:textAllCaps">true</item>
    <item name="android:letterSpacing">0.11</item>
    <item name="android:textColor">@color/sdk_primary_inverted_color</item>
</style>
```

- Subhead

```
<style name="sdk_subhead">
    <item name="android:fontFamily">sans-serif-medium</item>
    <item name="android:textSize">18sp</item>
    <item name="autoSizeTextType">uniform</item>
    <item name="autoSizeMaxTextSize">18sp</item>
    <item name="android:textAllCaps">false</item>
    <item name="android:textColor">@color/sdk_text_value_color</item>
</style>
```

- Copytext

```
<style name="sdk_copytext">
    <item name="android:fontFamily">sans-serif</item>
    <item name="android:textSize">16sp</item>
    <item name="autoSizeTextType">uniform</item>
    <item name="autoSizeMaxTextSize">18sp</item>
    <item name="android:textColor">@color/sdk_text_value_color</item>
</style>
```

- Copytext Bold

```
<style name="sdk_bold_copytext">
```

```
        <item name="android:fontFamily">sans-serif</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textSize">16sp</item>
        <item name="android:textColor">@color/sdk_text_value_color</item>
    </style>
```

- Copytext Inverted

```
    <style name="sdk_inverted_copytext">
        <item name="android:fontFamily">sans-serif</item>
        <item name="android:textSize">16sp</item>
        <item name="android:textColor">@color/sdk_primary_inverted_color</item>
    </style>
```

- Card Title

```
    <style name="sdk_card_title">
        <item name="android:fontFamily">sans-serif-medium</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textSize">16sp</item>
        <item name="autoSizeTextType">uniform</item>
        <item name="autoSizeMaxTextSize">18sp</item>
        <item name="android:textAllCaps">true</item>
        <item name="android:letterSpacing">0.1</item>
        <item name="android:textColor">@color/sdk_primary_color</item>
    </style>
```

- Notification Banner Text

```
    <style name="sdk_notification_banner">
        <item name="android:fontFamily">sans-serif</item>
        <item name="android:textSize">14sp</item>
        <item name="android:textColor">
@color/sdk_icon_notification_banner_color</item>
    </style>
```

- Error Banner Text

```
    <style name="sdk_error_banner">
        <item name="android:fontFamily">sans-serif</item>
        <item name="android:textSize">14sp</item>
        <item name="android:textColor">@color/sdk_icon_error_banner_color</item>
    </style>
```

- Overview Insured

```xml
<style name="sdk_card_overview_insured">
    <item name="android:fontFamily">sans-serif-medium</item>
    <item name="android:textSize">18sp</item>
    <item name="android:textAllCaps">false</item>
    <item name="android:textColor">@color/sdk_text_value_color</item>
</style>
```

- Overview Expiry Date

```xml
<style name="sdk_card_overview_expirydate">
    <item name="android:fontFamily">sans-serif</item>
    <item name="android:textSize">13sp</item>
    <item name="android:textAllCaps">false</item>
    <item name="android:textColor">@color/sdk_text_label_color</item>
</style>
```

- Overview Expired

```xml
<style name="sdk_card_overview_expired">
    <item name="android:fontFamily">sans-serif</item>
    <item name="android:textSize">13sp</item>
    <item name="android:textAllCaps">false</item>
    <item name="android:textColor">@color/sdk_error_color</item>
</style>
```

- Insured

```xml
<style name="sdk_card_insured">
    <item name="android:fontFamily">sans-serif-medium</item>
    <item name="android:textSize">18sp</item>
    <item name="android:textAllCaps">false</item>
    <item name="android:textColor">@color/sdk_text_value_color</item>
</style>
```

- Policy Number

```xml
<style name="sdk_card_insured_policy">
    <item name="android:fontFamily">sans-serif-light</item>
    <item name="android:textSize">16sp</item>
    <item name="android:letterSpacing">0.05</item>
    <item name="android:textAllCaps">false</item>
    <item name="android:textColor">@color/sdk_text_value_color</item>
</style>
```

- Label

```
    <style name="sdk_card_front_label">
        <item name="android:fontFamily">sans-serif</item>
        <item name="android:textSize">13sp</item>
        <item name="android:textAllCaps">false</item>
        <item name="android:textColor">@color/sdk_text_label_color</item>
    </style>
```

- Value

```
    <style name="sdk_card_front_value">
        <item name="android:fontFamily">sans-serif-medium</item>
        <item name="android:textSize">16sp</item>
        <item name="android:textAllCaps">false</item>
        <item name="android:textColor">@color/sdk_text_value_color</item>
    </style>
```

- Complementary Card Back Freefield

```
    <style name="sdk_card_back_complementary_free_field">
        <item name="android:fontFamily">sans-serif-mediumm</item>
        <item name="android:textSize">18sp</item>
        <item name="android:textAllCaps">false</item>
        <item name="android:textColor">@color/sdk_text_value_color</item>
    </style>
```

- 1st Level Button Text

```
    <style name="sdk_btn_primary_text">
        <item name="android:fontFamily">sans-serif-medium</item>
        <item name="android:textSize">16sp</item>
        <item name="android:textAllCaps">false</item>
    </style>
```

- 1st Level Button Inverted Text

```
    <style name="sdk_btn_primary_inverted_text">
        <item name="android:fontFamily">sans-serif-medium</item>
        <item name="android:textSize">16sp</item>
        <item name="android:textAllCaps">false</item>
    </style>
```

- 2nd Level Button Text

```
    <style name="sdk_btn_secondary_text">
```

```
        <item name="android:fontFamily">sans-serif-medium</item>
        <item name="android:textSize">16sp</item>
        <item name="android:textAllCaps">false</item>
    </style>
```

- 2nd Level Button Inverted Text

```
    <style name="sdk_btn_secondary_inverted_text">
        <item name="android:fontFamily">sans-serif-medium</item>
        <item name="android:textSize">16sp</item>
        <item name="android:textAllCaps">false</item>
    </style>
```

Other Textviews

Texts, which are not listed in Textview Styles, are outside of the Design Customization, and will use the system font of the device. However, if the theme of the host app overwrites the fontFamily with another font, this font is used also here.

The textviews are optimized for the Roboto-FontFamily. To keep those texts in its system font, ensure that the fontFamily of `Theme.AppCompat.Light` is not affected by the host app.

Style Items

- FontFamily

  There are following fontFamilies possible to be set in the default Android Font: `serif, serif-monospace, sans-serif, sans-serif-light, sans-serif-medium, sans-serif-black, sans-serif-thin, sans-serif-condensed, sans-serif-condensed-light, sans-serif-condensed-medium, sans-serif-smallcaps`

  Or store a specific font as ttf-File `(my_font.ttf)` under the `/res/values` folder of your application and refer to it this way:

```
    <item name="android:fontFamily">@font/my_font</item>
```

- TextStyle

  TextStyle can be set to `bold`, `italic` or `normal`. If not defined, style will be `normal`.

- TextSize

  TextSizes should be defined in `sp (scale-independent pixels)` and it's recommended to keep them between `13sp` and `18sp`.

- TextColor

  TextColors can be defined as Hex Codes, or as reference by name to another existing defined color.

```
    <color name="name_of_color_white_as_hex_code">#FFFFFF</color>
    <color name="name_of_color_same_as_another_color">
@color/name_of_another_color</color>
```

Colors

Follow the steps below in order to customize the colors of the VIC SDK:

1. Create a new resource file under the `/res/values` folder of your application or open an already existing resource file.
2. Copy-Paste the color elements you want to change from below.
3. Overwrite the value with a color hexa-value of your choice or a reference to an already existing color.

Default Colors

Overwrite following color values to change the color theme through the whole sdk.

```
    <color name="sdk_primary_color">#0367B1</color>
    <color name="sdk_primary_color_dark">#0246A1</color>
    <color name="sdk_primary_color_light">#E1F3FD</color>
    <color name="sdk_primary_inverted_color">#FFFFFF</color>
    <color name="sdk_text_label_color">#696969</color>
    <color name="sdk_text_value_color">#000000</color>
    <color name="sdk_text_value_disabled_color">@color/sdk_text_label_color</color>
    <color name="sdk_transparent">@android:color/transparent</color>
    <color name="sdk_error_color">#CF061D</color>
    <color name="sdk_disabled_background_color">#C0C0C0</color>
```

Buttons

There are 4 Different Button types and all have states `enabled`, `selected`, or `disabled`. To each state per button the text color, background color and border color can be changed individually.

- Primary Button

```
    <color name="sdk_btn_primary_normal_text_color"
>@color/sdk_primary_inverted_color</color>
    <color name="sdk_btn_primary_selected_text_color"
>@color/sdk_primary_inverted_color</color>
    <color name="sdk_btn_primary_disabled_text_color"
>@color/sdk_btn_disabled_text_color</color>
    <color name="sdk_btn_primary_normal_background_color">
@color/sdk_primary_color</color>
    <color name="sdk_btn_primary_selected_background_color"
>@color/sdk_primary_color_dark</color>
```

```
    <color name="sdk_btn_primary_disabled_background_color"
>@color/sdk_btn_disabled_background_color</color>
    <color name="sdk_btn_primary_normal_border_color">
@color/sdk_primary_color</color>
    <color name="sdk_btn_primary_selected_border_color">
@color/sdk_primary_color</color>
    <color name="sdk_btn_primary_disabled_border_color"
>@color/sdk_btn_disabled_border_color</color>
```

- Primary Button Inverted

```
    <color name="sdk_btn_primary_inverted_normal_text_color">
@color/sdk_transparent</color>
    <color name="sdk_btn_primary_inverted_selected_text_color">#33000000</color>
    <color name="sdk_btn_primary_inverted_disabled_text_color"
>@color/sdk_btn_disabled_text_color</color>
    <color name="sdk_btn_primary_inverted_normal_background_color"
>@color/sdk_primary_inverted_color</color>
    <color name="sdk_btn_primary_inverted_selected_background_color"
>@color/sdk_disabled_background_color</color>
    <color name="sdk_btn_primary_inverted_disabled_background_color"
>@color/sdk_btn_disabled_background_color</color>
    <color name="sdk_btn_primary_inverted_normal_border_color"
>@color/sdk_primary_color</color>
    <color name="sdk_btn_primary_inverted_selected_border_color"
>@color/sdk_primary_color</color>
    <color name="sdk_btn_primary_inverted_disabled_border_color"
>@color/sdk_btn_disabled_border_color</color>
```

- Secondary Button

```
    <color name="sdk_btn_secondary_normal_text_color">
@color/sdk_primary_color</color>
    <color name="sdk_btn_secondary_selected_text_color"
>@color/sdk_primary_color_dark</color>
    <color name="sdk_btn_secondary_disabled_text_color"
>@color/sdk_btn_disabled_text_color</color>
    <color name="sdk_btn_secondary_normal_background_color"
>@color/sdk_primary_inverted_color</color>
    <color name="sdk_btn_secondary_selected_background_color"
>@color/sdk_primary_color_light</color>
    <color name="sdk_btn_secondary_disabled_background_color"
>@color/sdk_btn_disabled_background_color</color>
    <color name="sdk_btn_secondary_normal_border_color">
@color/sdk_primary_color</color>
    <color name="sdk_btn_secondary_selected_border_color">
@color/sdk_primary_color</color>
    <color name="sdk_btn_secondary_disabled_border_color"
```

```
>@color/sdk_btn_disabled_border_color</color>
```

- Secondary Button Inverted

```
    <color name="sdk_btn_secondary_inverted_normal_text_color"
>@color/sdk_primary_inverted_color</color>
    <color name="sdk_btn_secondary_inverted_selected_text_color"
>@color/sdk_primary_inverted_color</color>
    <color name="sdk_btn_secondary_inverted_disabled_text_color"
>@color/sdk_btn_disabled_text_color</color>
    <color name="sdk_btn_secondary_inverted_normal_background_color"
>@color/sdk_transparent</color>
    <color name="sdk_btn_secondary_inverted_selected_background_color">
#33000000</color>
    <color name="sdk_btn_secondary_inverted_disabled_background_color"
>@color/sdk_btn_disabled_background_color</color>
    <color name="sdk_btn_secondary_inverted_normal_border_color"
>@color/sdk_primary_inverted_color</color>
    <color name="sdk_btn_secondary_inverted_selected_border_color"
>@color/sdk_primary_inverted_color</color>
    <color name="sdk_btn_secondary_inverted_disabled_border_color"
>@color/sdk_btn_disabled_border_color</color>
```

- Default Colors for Disabled Button

If not already overwritten by other individual colors, the colors for the disabled states of all buttons can be changed by overwriting following values:

```
    <color name="sdk_btn_disabled_text_color">@color/sdk_text_label_color</color>
    <color name="sdk_btn_disabled_background_color"
>@color/sdk_disabled_background_color</color>
    <color name="sdk_btn_disabled_border_color">
@color/sdk_disabled_background_color</color>
```

Cards

The border and background colors for the front card view can be changed by overwriting following values:

```
    <color name="sdk_card_background_color">@color/sdk_primary_inverted_color</color>
    <color name="sdk_card_inner_background_color">
@color/sdk_primary_color_light</color>
    <color name="sdk_card_border_color">#CACACA</color>
    <color name="sdk_card_inner_border_color">#33000000</color>
```

Pills (Pagination)

The active and passive pills colors can be changed by overwriting following values:

```
    <color name="sdk_pill_active">@color/sdk_primary_color</color>
    <color name="sdk_pill_passive">@color/sdk_disabled_background_color</color>
```

Scrollbar

The scrollbar color can be changed by overwriting following value:

```
    <color name="sdk_scrollbar">@color/sdk_primary_color</color>
```

Banner

The banner background colors can be changed by overwriting following values:

```
    <color name="sdk_notification_banner_background_color">
@color/sdk_primary_color</color>
    <color name="sdk_error_banner_background_color">@color/sdk_error_color</color>
```

Icons

The Icon colors can be changed by overwriting following values:

```
    <color name="sdk_icon_close_color">@color/sdk_text_value_color</color>
    <color name="sdk_icon_info_color">@color/sdk_primary_color</color>
    <color name="sdk_icon_flip_color">@color/sdk_primary_color</color>
    <color name="sdk_icon_notification_banner_color">
@color/sdk_primary_inverted_color</color>
    <color name="sdk_icon_error_banner_color">
@color/sdk_primary_inverted_color</color>
    <color name="sdk_icon_system_color">@color/sdk_primary_color</color>
    <color name="sdk_icon_system_inverted_color">
@color/sdk_primary_inverted_color</color>
```

Dimensions

For each Button type and the front cards, the border width and radius can be changed indivitually. Follow the steps below in order to customize those values in the VIC SDK:

1. Create a new resource file under the `/res/values` folder of your application or open an already existing resource file.
2. Copy-Paste the dimension elements you want to change from below.

3. Overwrite the value with a dp-Value of your choice or a reference to an already existing dimension.

```xml
<resources>
    <!--    DEFAULT BUTTON DIMENSIONS-->
    <dimen name="sdk_btn_border_radius">5dp</dimen>
    <dimen name="sdk_btn_border_width">1dp</dimen>
    <dimen name="sdk_no_stroke">0dp</dimen>

    <!--    PRIMARY BUTTON DIMENSIONS-->
    <dimen name="sdk_btn_primary_border_radius">@dimen/sdk_btn_border_radius</dimen>
    <dimen name="sdk_btn_primary_border_width">@dimen/sdk_no_stroke</dimen>

    <!--    PRIMARY INVERTED BUTTON DIMENSIONS-->
    <dimen name="sdk_btn_primary_inverted_border_radius">
@dimen/sdk_btn_border_radius</dimen>
    <dimen name="sdk_btn_primary_inverted_border_width">@dimen/sdk_no_stroke</dimen>

    <!--    SECONDARY BUTTON DIMENSIONS-->
    <dimen name="sdk_btn_secondary_border_radius">@dimen/sdk_btn_border_radius</dimen>
    <dimen name="sdk_btn_secondary_border_width">@dimen/sdk_btn_border_width</dimen>

    <!--    SECONDARY INVERTED BUTTON DIMENSIONS-->
    <dimen name="sdk_btn_secondary_inverted_border_radius"
>@dimen/sdk_btn_border_radius</dimen>
    <dimen name="sdk_btn_secondary_inverted_border_width">
@dimen/sdk_btn_border_width</dimen>

    <!--    DISABLED BUTTON DIMENSIONS-->
    <dimen name="sdk_btn_disabled_border_radius">@dimen/sdk_btn_border_radius</dimen>
    <dimen name="sdk_btn_disabled_border_width">@dimen/sdk_btn_border_width</dimen>

    <!--    CARD BORDER DIMENSIONS-->
    <dimen name="sdk_card_border_radius">6dp</dimen>
    <dimen name="sdk_card_border_width">1dp</dimen>

    <dimen name="sdk_card_inner_border_radius">9dp</dimen>
    <dimen name="sdk_card_inner_border_shadow_radius"
>@dimen/sdk_card_inner_border_radius</dimen>
    <dimen name="sdk_card_inner_border_width">1dp</dimen>

    <dimen name="sdk_card_small_inner_border_radius">4dp</dimen>
    <dimen name="sdk_card_small_inner_border_shadow_radius"
>@dimen/sdk_card_small_inner_border_radius</dimen>
    <dimen name="sdk_card_small_inner_border_width" format="float">0.5dp</dimen>
</resources>
```

Card backgrounds

Add any custom card backgrounds to your `/res/drawable` folder, for example `res/drawable/my_card_background.png`. This image has to be a PNG with the bottom part empty as the card-holder information is put on there dynamically. You can set the default card background after initializing the SDK:

```
override fun onCreate() {
    super.onCreate()
    VicSdk.init(this)
        .setCardImage(R.drawable.my_card_background)
}
```

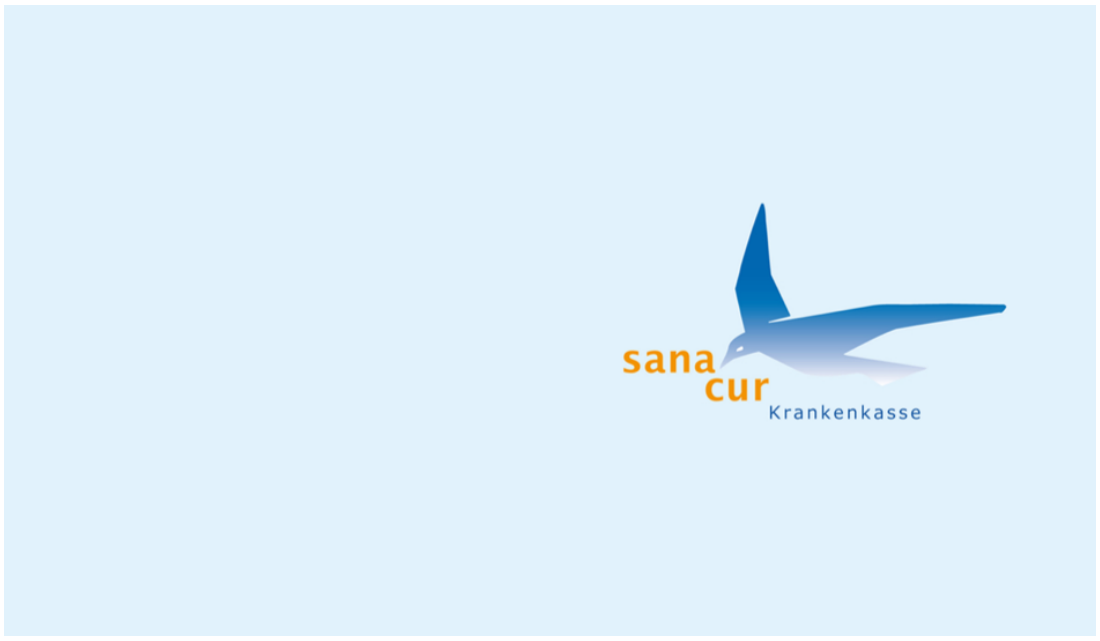
To have individual designs for complementary and mandatory cards, the type such as `VicCardType.MANDATORY` can be specified:

```
VicSdk.setCardImage(R.drawable.my_card_background, VicCardType.MANDATORY)
}
```

To have individual designs for specific insured persons, their card number as String without spaces can be passed also:

```
VicSdk.setCardImage(R.drawable.my_card_background, cardNumber =
"80756012340000000108")
}
// or
VicSdk.setCardImage(R.drawable.my_card_background, VicCardType.COMPLEMENTARY_BACK,
"80756012340000000108")
}
```

1. If a custom image exists for a card number and type, this image will be used.
2. Else, if a custom image for a card number without specified type exists, this image will be used. (Except for `VicCardType.COMPLEMENTARY_BACK`, this type has higher priority than the card number)
3. Else, if a custom image for the type without specified card number exists, this image will be used.
4. Otherwise, the default background will be used.



Front Side backgrounds

The card image should have a form factor of 1:1.73 e.g.: 945x546 pixel.

The logo is to be placed in the right half of the image

The logo must be horizontally right aligned and vertically centered.

The maximum size of the logo is 70% width and 60% height of the total image.

For complementary cards *(Zusatzversicherungen)* the type `VicCardType.COMPLEMENTARY` should be used.

```
VicSdk.setCardImage(R.drawable.my_card_background, VicCardType.COMPLEMENTARY)
```

Back Side backgrounds

For complementary cards a specific background image can be set for the back side.

The image will be rotated by 90° and stretched uniformly so that both dimensions (width and height) of the image will be equal to or larger than the corresponding card dimension. The card dimension and its ratio can vary according to device.

For this background the type `VicCardType.COMPLEMENTARY_BACK` should be used.

```
VicSdk.setCardImage(R.drawable.my_card_background, VicCardType.COMPLEMENTARY_BACK)
```

The default will be a white background.

Card Text Color

The default text color inside of a card is controlled by the `sdk_text_value_color`. It can be overwritten for specific cards in a similar way like the Card background settings. The color of the text on top of the card (next to the country flag) is set via text color. The color of the text on the bottom of the card is set via secondary text color. If the secondary text color is not specified, the one specified in text color is used.

```
VicSdk.setCardTextColor("#FFFFFF", VicCardType.COMPLEMENTARY)
VicSdk.setCardSecondaryTextColor("#ABCDEF", cardNumber = "YOUR_CARD_NUMBER")
```

Card’s free fields

Mandatory Card’s free fields

Copy the following style element into a resource file under `/res/values` and replace YOUR_FREE_TEXT with the text of your choice. A line break can be put by inserting `\n` in it.

```
    <string name="sdk_card_back_free_field" translatable="false">
YOUR_FREE_TEXT</string>
```

The text set in the resource file will be used for all mandatory cards. It is also possible to override the mandatory free field for a specific card number:

```
VicSdk.setMandatoryFreeField("YOUR_FREE_TEXT", cardNumber = "80756012340000000108")
```

Complementary Card’s free fields

Complementary card’s text on the back side can be changed in the same way by code `"sdk_card_back_complementary_free_field"`.

```
    <string name="sdk_card_back_complementary_free_field" translatable="false"
>YOUR_FREE_TEXT</string>
```

Tutorial-Screens

By default on clicking the info button on the frontside of a card there are five screen shown in a sequence:

1. `TutorialScreen.WELCOME_SCREEN`
2. `TutorialScreen.QR_CODE_SCREEN`
3. `TutorialScreen.BARCODE_SCREEN`

4. `TutorialScreen.NFC_SCREEN`
5. `TutorialScreen.BACK_OF_CARD_SCREEN`

By choice screens can be hidden with following function:

```
VicSdk.hideTutorialScreen(
    TutorialScreen.WELCOME_SCREEN,
    TutorialScreen.BACK_OF_CARD_SCREEN,
    TutorialScreen.QR_CODE_SCREEN
)
```

If all screen are hidden, also the info button will be hidden.

If showComplementaryCardBackside is set to false, the TutorialScreen.BACK_OF_CARD_SCREEN will also be automatically hidden. This only applies for complementary cards, for all other cards when clicking the info button, the TutorialScreen.BACK_OF_CARD_SCREEN will still be shown (unless it was hidden app-wide by VicSdk.hideTutorialScreen(TutorialScreen.BACK_OF_CARD_SCREEN))

Card-Data

On the Front Card, by defaultcard data attributes in following order are shown:

```
CARD_NUMBER, BAG_NUMBER, AHV_NUMBER, BIRTHDATE, EXPIRY_DATE, BARCODE
```

The visibility and order of those attributes can be changed by redefining the `VicSdk.cardListItems` -List

```
VicSdk.cardListItems = arrayListOf(CardListItem.BARCODE, CardListItem.AHV_NUMBER)
// this would show the Bardcode and the AHV Number below the inner Card
```

Complementary Card Backside

Showing the backside of complementary cards can be enabled/disabled (per insurance not per card). Default: enabled. To disable use

```
VicSdk.setShowComplementaryCardBackside(false)
```

This disables the rotation when clicking on the card frontside and also hides the rotation button

Card Backside English

The european backside of cards can be shown in English regardless of the language set for the app. Default: false. To set English as default for the backside use

```
VicSdk.enableEnglishEuropeanCard(true)
```

Policy Number Label Alternative

The label of the policy number can show an alternative text. The alternative texts are as following:

- EN: Insured person no.
- DE: Versicherten-Nr.
- FR: N° d’assuré(e)
- IT: Numero d’assicurato

Default: false. To use the alternative labels use

```
VicSdk.enableAlternativeLabelForPolicyNumber(true)
```