Twigbit Ident SDK

Status	Version
DRAFT	0.1.0 unreleased

NOTICE: This is a confidential document.

The Twigbit Ident SDK is a lightweight convenience layer on top of the AusweisApp2 SDK written in Kotlin. We are aiming to extract and eliminate the recurring code and configuration that every developer faces integrating the SDK.

Features

- Simplify the tedious AusweisApp2 SDK configuration
- Replace the JSON based messanging system by convenient wrapper methods, giving developers to must-have convenience such as code completion
- Lightweight- besides the AusweisApp2 SDK, the only other dependency is Google GSON for JSON parsing
- Drop-in UI - Provide a simple, customizable drop in UI as a quick integration with identification processes
- (coming soon) Capability check- check whether the users device has the required architecture and NFC capabilities
- (coming soon) A custom identification app as a zero-dependency option for the integration
- (coming soon) Provides a fallback to prompt the user to install the official [AusweisApp2] (https://www.ausweisapp.bund.de/) in case of unsupported architecture (see limitations below)

Limitations

- The AusweisApp2 SDK only supports arm64-v8a architecures since version 15.03. Unfortunalety, we are bound to that limitation.

Usage

The usage examples are provided in Kotlin. The integration works in Java analogously.

Download

To get access to the SDK, please get in touch.

Gradle:

```
dependencies {
  implementation 'com.twigbit.identsdk:identsdk:1.0.0'
}
```

Maven:

```
<dependency>
  <groupId>com.twigbit.identsdk</groupId>
  <artifactId>identsdk</artifactId>
  <version>1.0.0</version>
</dependency>
```

Option 1: Identify users with the Drop-In UI (alpha)

To get started quickly and have the SDK take care of the entire identification process for you, you can use the build-in Drop-in UI.

To start an identification process, simply create a `DropInRequest` with your servers tcTokenURL and start the activity for the result.

```
val REQUEST_CODE_IDENTIFICATION = 0

private fun startDropInIdentification(){
    val dropInRequest = DropInRequest("https://...") // your tcToken
Endpoint
    startActivityForResult(dropInRequest.getIntent(this),
REQUEST_CODE_IDENTIFICATION)
}
```

To receive the identification result, you should override your activities `onActivityResult`.

```
    override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
        if (requestCode == REQUEST_CODE_IDENTIFICATION) {
            if (resultCode == Activity.RESULT_OK) {
                // Success. Update the UI to reflect the successful
identification
                // and fetch the user data from the server where they were
delivered.
            } else if (resultCode == Activity.RESULT_CANCELED) {
                // The user canceled the identification
            } else {
                // An error occured during the identification
            }
        }
    }
```

Option 2: Implement your own UI

To receive the SDK's identification state callbacks in your activity, extend the `IdentificationActivty`. This will bind an `IdentificationManager` instance to your activities lifecycle.

To receive the identification state events, you must implement the `onStateChanged` method. As this method might be called from a different thread, be sure to run all UI operations on your UI thread explicity.

```
class MainActivity : IdentificationActivity() {
    // TODO @dev rename to `onIdentificationStateChanged`
    override fun onStateChanged(state: String, data: String?) {
        runOnUiThread {
            when (state) {
                IdentificationManager.STATE_COMPLETE -> {
                    // The identification was complete, display a success
message to the user and fetch the identification result from the server
                }
                IdentificationManager.STATE_ACCESSRIGHTS -> {
                    // A list of the id-card fields that the sdk is trying
to access has arrived. Display them to the user and await his
confirmation.
                    // TODO @dev continue with runIdent(), this was
treated non-blocking until now.
                    // TODO @dev better parameter typing
                }
                IdentificationManager.STATE_CARD_INSERTED -> {
                    // A card was attached to the NFC reader
                    // TODO @dev show empty card and detach data.
                }
                IdentificationManager.STATE_ENTER_PIN -> {
                    // The id cards PIN was requested. Display a PIN
dialog to the user.
                    // To continue the identification process, call
identificationManager.setPin(pin: String)
                }
                IdentificationManager.STATE_ENTER_PUK -> {
                    // The id cards PUK was requested. Display a PUK
dialog to the user.
                    // To continue the identification process, call
identificationManager.setPuk(puk: String)
                }
                IdentificationManager.STATE_ENTER_CAN -> {
                    // The id cards CAN was requested. Display a CAN
dialog to the user.
                    // To continue the identification process, call
identificationManager.setCan(can: String)
                }
                IdentificationManager.STATE_BAD -> {
                    // Bad state. Display an error/issue dialog to the
user.
                    // TODO @dev figure out reasons for bad state, offer
solutions, i.e. id card blocked, id card detached. More granular apporach
needed.
                }
            }
        }
```

```
    }
}
```

Start the identification flow by calling the `IdentificationManager.startIdent` Method from your `IdentificationActivity`. The `identificationManager` object is accessible through your parent activity.

```
fun onStartIdent(v: View) {
  identificationManager.startIdent("https://...") // TODO("Your tcTokenURL here")
}
```

Usage with AusweisIdent (alpha)

This SDK provides useful helpers, if you are planing to use AusweisIdent offered by the Bundesdruckerei GmbH and Governikus KG.

In order to use AusweisIdent you need to provide the Ident SDK with a tcTokenURL pointing to an AusweisIdent server.

```
val ausweisIdentTcTokenUrl = AusweisIdentBuilder()
        .scope(AUSWEISIDENT_SCOPE_FAMILYNAMES)
        .scope(AUSWEISIDENT_SCOPE_PLACEOFBIRTH)
        .clientId("ABCDEFG")
        .redirectUrl("https://yourserver.com")
        .build()
```

Get raw JSON Messanges (alpha)

If you want to get access to the AusweisApp2 SDK's string messages, implement the `onMessage` callback. This should only be used for debugging purposes.

```
class MainActivity : IdentificationActivity() {

    ...

    override fun onMessage(message: Message) {
        // Log the message, etc.
    }
}
```

Sample

A working implementation can be found in the `/samples` directory. Please note that you need a test PA to test the identification flow in the reference system.

0-dependency integration (coming soon) with dedicated Identification App

Backlog / Nice to have

- Vibrate on message (configuration option necessary?)

Copyright