Developer's Guide for Android

The **IDEMIA Verify SDK - Android** v1.4.1 is targeted at developers who want to integrate the ability to verify a **Mobile ID** credential (mID), also known as, a mobile driver's license (mDL) in an existing mobile app.

Prerequisites

Skills Required

The integration tasks should be done by developers with knowledge of:

- Android Studio
- Java/Kotlin for Android
- Android operating system

Resources Required

Integrations may be performed on PC Windows, Linux, or Macintosh.

The tools required are:

- Android Studio 4 or above
- Android SDK tools: preferred latest version (release 24 or above)
- JDK: preferred latest version (7 or above)
- Android device (emulator is not supported)
- Minimum SDK version is 21 (Android 5.0)

Tech Stack

Tech	Description
DEVELOPMENT ENVIRONMENT:	Android Studio 4.0+, Windows, Mac or Linux, JDK 8.. JDK12
PROGRAMMING LANGUAGE:	Kotlin 1.3.71, Java 8... 12
RUN TIME:	Android 5.0 (API 21) or greater
PDF417 SCANNING SDK:	Microblink 7.1.0
QR CODE SCANNING SDK:	Microblink 7.1.0
OPTICAL SCANNING (BIOMETRIC TEMPLATE):	Biometric Capture SDK 4.23.1
LICENSE (LKMS):	Required for Optical Inspection BioTemplate authentication

SDK APIs and Methods

The SDK is made up of following components:

`IDVerifyTerminal` acts as the entry point for initializing device engagement, sending the request to the **Mobile ID** credential holder, receiving the response, and sending the parsed model back to the calling app. It exposes the following APIs to caller/client with the help of the concrete class `IDVerifyClient`.

- version

```
  //Use IDVerifyTerminal.version to read Verify SDK version
  Log.d(TAG, "SDK Version=${IDVerifyTerminal.version}")
```

- parsePDF417

```
/** Parse the AAMVA raw data retrieved from physical or mobile driving license
 * @param String: raw/text data scanned from the 2D/PDF417 Barcode
 * @param callback: callback to return the response model*/
  abstract fun parsePDF417(apiConfigs: ApiConfigs?, callback:
ResponseCallback<in ResponseItem?>)
```

- initDeviceEngement

```
/**
 * Parse the device engagement data received from QR code for NFC
 * @param String: raw/text data scanned from QR code/Barcode etc.
 * @param callback: callback to return the response model
 * */
 abstract fun initDeviceEngagement(apiConfigs: ApiConfigs?, callback:
ResponseCallback<in ResponseItem?>)
```

- sendRequest

```
/**
 * sends request to the IDEMIA Mobile ID App based on the use case selected by
the user in IDEMIA Mobile ID Verify App
 * @param callback: callback to send progress and response data back to app
 *
 * 1. create a Mobile ID credential request based on the supplied use case or
parameters/configs
 * 2. onCreate communication channel (BLE/NFC/wifiAware, etc...) as per
engagement data
 * 3. send the request to the IDEMIA Mobile ID  App on the communication channel
 *
 * Note: In the future we may also use USECASE: use case identifier required to
prepare desired request data
 */
abstract fun sendRequest(apiConfigs: ApiConfigs? = ApiConfigs.Builder().build(),
                         callback: ICallback<in ResponseItem?, in ProgressItem>)
```

- cancelRequest

```
/**
 *   cancels any transaction/running session with a Mobile ID credential.
Overloads to default cancellation
 */
open fun cancelRequest(): Boolean {
    return cancelRequest(CancellationReason.CANCELED_BY_USER)
}
```

- generatePDF417

```
/**
 * Rendering PDF417 barcode
 * @param ApiConfigs
 * @param ResponseCallback<in ResponseItem?>
 * */
    abstract fun generatePDF417(apiConfigs: ApiConfigs, callback:
ResponseCallback<in ResponseItem?>)
```

Design Considerations

End-user permissions have to be handled by the integrator/app. Specifically, you will need to check that the app permissions are granted by the end-user if the Android version is higher than 23 (Android 6.0 Marshmallow). The SDK will throw an exception or error if the required permissions are not granted.

1. Camera permission (for QR code scan)
2. Location permission (only when BLE client mode is used)

Sample Project

To get a copy of the **IDEMIA Verify Sample App – Android**, please speak to your local Sales Manager.

How to Create Your Own App

To create your own app, you must:

1. **Add the SDK libraries to your project.** There are two ways of doing this: Option A or Option B.

Option A: Import the SDK as .aar

1. Copy the idverifysdk-1.4.1.aar into your app/libs directory.
2. Gradle sync and build.
3. Add the below line in app/build.gradle.

```
dependencies {
   implementation fileTree(include: ['*.jar', '*.aar'], dir: 'libs')
   ...
 }
```

Option B: Import the SDK from Artifactory

1. Add the following configs in gradle.properties:

```
artifactory_remoteUrl=https://mi-artifactory.otlabs.fr/artifactory/verify-sdk-android-release
artifacto ry_username=<username>
artifactory_password=<password>
```

2. Add the following dependency in your project/build.gradle:

```
buildscript {
  repositories {
      google()
      jcenter()
      maven {
          url "https://plugins.gradle.org/m2/"
      }
      maven {
          url "${artifactory_remoteUrl}"
          credentials {
              username = "${artifactory_user}"
              password = "${artifactory_password}"
          }
      }
  }
  ...
}
...
allprojects {
     repositories {
        ...
        maven { url 'https://github.com/c-rack/cbor-java' }
        maven {
            url "${artifactory_remoteUrl}"
            credentials {
                username = "${artifactory_username}"
                password = "${artifactory_password}"
            }
        }
    }
    project.ext{
        sdkVersionName='1.4.1'
        sdkVersionCode=xxxx //latest release version number
        cborLibsVersion='0.9'
        bouncyCastleVersion='1.60'
        zxingVersion='3.6.0'
        zxingCoreVersion='3.3.2'
    }
  }
```

3. Add following dependency in your app/build.gradle:

```
     dependencies {
     ....
     implementation
("com.idemia.idverify.android:${project.ext.sdkVersionName}:${project.ext.sdkVer
sionCode}@aar") {
        transitive = true
    }
}
```

4. Include the supporting dependencies in app/build.gradle:

The following supporting dependencies needs to be added in app/build.gradle for the **IDEMIA Mobile ID Verify App** to work.

```
    dependencies {
    //additional dependencies required for idemia id.verify sdk
     implementation "co.nstant.in:cbor:${project.ext.cborLibsVersion}"
     implementation "org.bouncycastle:bcprov-
jdk15on:${project.ext.bouncyCastleVersion}"
     implementation("com.journeyapps:zxing-android-
embedded:${project.ext.zxingVersion}") { transitive = false }
     implementation "com.google.zxing:core:${project.ext.zxingCoreVersion}"
  }
```

5. Include additional dependencies for the Optical Inspection feature (if used):

```
    dependencies {
        ...
      //#######libs needs to be added in integrator app #####//
        implementation('morpho.mph_bio_sdk.android:SmartBio:4.23.1@aar') {
            transitive = true
        }
        implementation "com.idemia.smartsdk:plugin-face-normal:4.23.1@aar"
        implementation 'com.idemia.smartsdk:plugin-algorithm-f5-4-
low75:4.23.1@aar'
        implementation 'net.zetetic:android-database-sqlcipher:3.5.9@aar'
        implementation 'morpho.lkms-android:lkms_core:3.0.2@aar'
        implementation 'morpho.lkms-android:service_provider_local:1.2.5@aar'
        implementation("org.bouncycastle:bcpkix-
jdk15on:${project.ext.bouncyCastleVersion}")
     //#######libs needs to be added in integrator app #####//
    }
```

6. For the **Line Code SDK** integration, please refer to the Line Code SDK Documentation.
7. Gradle sync and build.
8. Add the required permissions to the project:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Integrating the QR Code and BLE Mode

1. Initialize the configurations of the **IDEMIA Verify SDK - Android** by calling `Configs.Builder().build()` in your activity/fragment.
2. Create the **IDEMIA Verify SDK - Android** client object to call the SDK APIs:

```
    val license = LicenseInfo(lkmsURL,lkmsProfileIdDev,lkmsApiKeyDev) //provide
a valid LKMS license details to VerifyID
    val config = VSDKConfigs.Builder(this)
                .setLicense(license) //optional; required only if you're to use
Optical Inspection
                .build()
    mSdkClient = IDVerifyTerminal.getClient(config)
```

Note:

- The minimum threshold for the BioTemplate authentication score (Optical Inspection use case) is 3500 by default.
- An internet connection is required for first-time transactions of Optical Inspection for attribute retrieval. Attributes are downloaded from the server only one time and will not be downloaded again until one or more attributes either expire or the **IDEMIA Mobile ID App** is reinstalled and or the data cleared.
- By default, no logs shall be visible on logcat from the SDK. Use enableDebug(true) for debugging purposes, only if needed, and toggle it to `false` for production release.

Use `DebugCallback` with `enableDebug` to get the logs via a callback and save them in file. This is useful in scenarios when logcat strips some logs because they are too big. e.g. response HEX with end-user image becomes too big to print in logcat.

```
val config = VSDKConfigs.Builder(this)
   .enableDebug(BuildConfig.DEBUG, object:DebugCallback{
    @override
    fun didReceiveLogs(logs: DLog){
        //receive logs.message, print or write to file
    }
   })
  ...
  .build()
```

- Enabling debug in the production app may adversely affect the communication speed and overall performance of the **IDEMIA Verify SDK - Android**. One simple way to do this is to use BuildConfig.DEBUG in place of using explicitly using true or false:

```
    enableDebug(BuildConfig.DEBUG)
```

3. Obtain a LKMS license.

Please contact our support team to get an LKMS license for your app. You can skip this step if your app doesn't intend to use the "Optical Inspection" feature of the **IDEMIA Mobile ID Verify App**.

4. Call the desired SDK API. Supply the necessary apiconfigs/request parameters and callback to receive the results.

- mSdkClient.initDeviceEngagment(apiconfigs,callback)

- mSdkClient.sendRequest(apiconfigs,callback)
- mSdkClient.cancelRequest()
  //so on...

5. Scan the QR code or PDF417 codes.

Follow the below steps to communicate with the **IDEMIA Mobile ID App** and display the results on the **IDEMIA Mobile ID Verify App** screen.

5.1 Capture the scan result in your activity or in the fragment `onActivityResult`. After scanning the QR code the result will display in the `onActivityResult` of the activity or fragment.

```
   override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        //handle all the results
        if (requestCode == REQUEST_SCAN_BARCODES_MICROBLINK) {
               if (resultCode == Activity.RESULT_OK) {
                  // updates bundled recognizers with results that have arrived
                  mRecognizerBundle.loadFromIntent(data)
                  val result = mBarcodeRecognizer.result
                  // *ScanType= QRCode //*ScanType= PDF417
                  Log.d(TAG, "ScanType= " + result.barcodeType.name)
                  doHandleScannedData(result.stringData, result.barcodeType)
                 }
            }
    }
```

5.2. Create the response callback for parsing engagement QR code data or PDF417.

```
 open fun doHandleScannedData(scannedData: String?, scanType: BarcodeType? =
null) {
        if (!scannedData.isNullOrEmpty()) {
            val apiConfigs = ApiConfigs.Builder()
                .withData(scannedData)
                .withImage(imageBytes or MorphoImage)//screenshot of other
device with user face image + QR code
                .build()
            when (scanType) {
                BarcodeType.QRCode -> {
                    mSdkClient?.initDeviceEngagement(apiConfigs,
mDEParserCallback)
                }
                BarcodeType.PDF417 -> {
                    mSdkClient?.parsePDF417(apiConfigs, m2DBarcodeParseCallback)
                }
                else -> {
                }
            }
        }
    }
  /** Response callback for parse engagement QR code data */
    private val mDEParserCallback = object :
ResponseCallback<DeviceEngagementModel?> {
        override fun onSuccess(deviceEngegmentModel: DeviceEngagementModel?) {
             // deviceEngegmentModel received. perform next step
             doStartDataTransfer(deviceEngegmentModel)
```

```
        }
        override fun onError(error: IError?) {
            handleError(error)
        }
    }
    /**or PDF417 raw data from physical or Mobile ID credential holder*/
    private val m2DBarcodeParseCallback = object : ResponseCallback<DLDataModel?
> {
        override fun onSuccess(result: DLDataModel?) {
             // PDF417 barcode scanning result. Display result screen
              displayResult(result)
        }
        override fun onError(error: IError?) {
            handleError(error as VSDKError)
        }
    }
```

6. Send the request to the **IDEMIA Mobile ID App** to get the credential details. The `sendRequest()` method will return the response from the end-user's **IDEMIA Mobile ID App**. If you've received it successfully, then you can display all the attributes of the **IDEMIA Mobile ID App** response in your own result activity.

```
    private fun doStartDataTransfer(deModel: DeviceEngagementModel?) {
       val namespaceParamList = getRequestedParamList() // make map of namespace
to List<MIDRequestParam>
       val apiConfigs = ApiConfigs.MIDRequestBuilder()
                   .version("1.0")
              //add docType
                   .addDocType(DOCTYPE_18013_5).apply {
                       addParams(NAMESPACE_18013_5, isoParamList)
                           .addParam(DOCTYPE_ORG_AAMVA_US,
MIDRequestParam(REAL_ID))
                           // pass all request params as Map<namespace-
>List<MIDRequest
                           .addParams(namespaceParamList)
                           //OR use another overload. duplicate params shall be
removed if found
                           .addParams(DOCTYPE_ORG_AAMVA_US,
namespaceParamList.get(DOCTYPE_ORG_AAMVA_US)!!)
                           .addParams(DOCTYPE_18013_5,
namespaceParamList.get(DOCTYPE_18013_5)!!)
                           //OR this way
                           .addParams(
                               "my.custom.namespace2", listOf(
                                   MIDRequestParam("custom_param_nm21"),
                                   MIDRequestParam("custom_param_nm22"),
                                   MIDRequestParam("custom_param_nm23"),
                                   MIDRequestParam("custom_param_nm21")
//duplicate
                               )
                           )
                           //OR use 3rd overload to pass params one by one
                           .addParam("my.custom.namespace",
MIDRequestParam("custom_param1"))
                           // add requestInfo as map
                           .addRequestInfo(
                               infoMap = hashMapOf(
```

```
                                    "reqKey1" to "reqVal1",
                                    "reqKey2" to "reqVal2",
                                    "reqKey3" to false,
                                    "reqKey4" to 3334
                                )
                            )
                            //or use other overload to add one by one
                            .addRequestInfo("reqinfo1", "value1")
                            .addRequestInfo("reqinfo2", "value2")
                            .addRequestInfo("reqinfoLong", 180L)
                            .addRequestInfo("reqinfoBool", true)
                            .addRequestInfo("reqinfoBytArr",
"sdkghdkgs".toByteArray())
                            .addRequestInfo("reqinfoDouble", 3.7)
                            .addRequestInfo("reqinfoFloat", 3.7F)
                            // add as dataItem
                            .addRequestInfo(
                                "reqinfoDataItem",

CborBuilder().addArray().add(1).add("item2").add("item3").end().build()
                                    .first()
                            )
                    }.buildDoc()
                //add another docType
                    .addDocType("my.cusom.docType.custom").apply {
                        addParams(
                            "my.custom.namespace2", listOf(
                                MIDRequestParam("custom_param_nm21", true),
                                MIDRequestParam("custom_param_nm22"),
                                MIDRequestParam("custom_param_nm23"),
                                MIDRequestParam("custom_param_nm21", true)
//duplicate
                            )
                        )
                            // add requestInfo as map
                            .addRequestInfo(
                                infoMap = hashMapOf(
                                    "reqKey1" to "reqVal1",
                                    "reqKey2" to "reqVal2",
                                    "reqKey3" to false,
                                    "reqKey4" to 3334
                                )
                            )
                    }.buildDoc()
                    .build()
            if (deModel?.isMDLAsPeripheral()) {
                    askLocationPermission() /*
android.permission.ACCESS_COARSE_LOCATION to be added in AndroidManifest */
                        //get permissin and then call sendRequest
             } else {
                    mSdkClient?.sendRequest(apiConfigs, callback =
mRequestCallback)
             }
        }
```

7. Obtain the response callback for parsing the engagement QR code data or the PDF417 scan:

```
  /**
     * Request callback for sendRequest SDK api to receive and display response
data from Mobile ID
     */
    val mRequestCallback = object : RequestCallback<ResponseItem?, ProgressItem>
{
        override fun onSuccess(response: ResponseItem?) {
            Log.e(TAG, "Mobile ID response:$response")
            displayResponse(response as DLDataModel)
        }
        override fun onError(error: IError?) {
            handleError(error = error as VSDKError)
        }
        override fun onProgress(progress: ProgressItem) {
            Log.e(TAG,  progress.toString())
        }
        override fun onRequestSent(status: Boolean, waitTimeMillis: Long) {
            super.onRequestSent(status, waitTimeMillis)
        }
    }
```

8. You can cancel a transaction in between any time if any error occurs in the app or if the end-user cancels the transaction by tapping the **Back** button. Cancel a transaction by calling the `cancelTransaction()` method.

```
    mSdkClient.cancelTransaction()
```

Deprecations/Deletions

- Deprecations in `DLDataModel`
- Direct access to `authStatus` field deprecated. Call `verificationStatus()` instead.

Additions/Updates

- Added optional method `withImage` in `ApiConfig.Builder` for `initDeviceEngagement` which takes the captured device screenshot image (while scanning a compact QR) for BioTemplate authentication
- Added optional method `setLicense` in `VSDKConfigs.Builder` to supply the LKMS license detail to the **IDEMIA Verify SDK - Android**
- Added model class `OIDLDataModel` which encapsulates the attributes list and corresponding security checks for the Optical Inspection QR code result
- Added new model class `OpticalInspectionAuth` to return security checks related to the Optical Inspection process.

```
    val security=responseModel.verificationStatus() as? OpticalInspectionAuth
    //security.isSuccess() returns true if all checks are successful
```

ProGuard Rules

The following rules should be added in the app proguard rules file:

- sdk obfuscation rules

```
-keep public class idemia.verify.sdk.** {
   public protected *;
}
```

- Additional rules to add if BioSDK/LKMS is used:

```
-keep class idemia.verify.sdk.biosdk.** { *; }
-keep class com.morpho.mph_bio_sdk.** { *; }
-keep class com.morpho.lkms.android.** {*;}
-keep interface morpho.sdk.* {*;}
-keep class net.sqlcipher.* {*;}
```

- Keep everything in this package from being removed or renamed:

```
-keep class morpho.urt.msc.models.** { *; }
```

- Keep everything in this package from being removed or renamed:

```
-keep class com.morpho.rt.** { *; }
```

```
-keep class morpho.rt.imageconvert.** { *; }
```

- Keep the class and specified members from being removed or renamed:

```
-keep class morpho.rt.imageconvert.MorphoImageConvert_JNI { *; }
```

If the above properties are not defined, LKMS will fail to resolve a valid license and the BioTemplate authentication will fail for the Optical Inspection use case.

Request parameters

Item #	Identifier	Param Name	CBOR Type
1	ADMINISTRATIVE_NUMBER	administrative_number	Major type 3
2	GENDER	gender	Major type 3
3	HEIGHT	height	Major type 0
4	WEIGHT	weight	Major type 0
5	EYE_COLOR	eye_color	Major type 3
6	HAIR_COLOR	hair_color	Major type 3
7	BIRTH_PLACE	birthplace	Major type 3
8	RESIDENT_ADDRESS	resident_address	Major type 3
9	PORTRAIT_CAPTURE_DATE	portrait_capture_date	Tag value 0 of major type 6
10	AGE_IN_YEARS	age_in_years	Major type 0
11	AGE_BIRTH_YEAR	age_birth_year	Major type 0
12	AGE_OVER_NN	age_over_NN	Value 20/21 of major type 7
13	ISSUING_JURISDICTION	issuing_jurisdiction	Major Type 3
14	NATIONALITY	nationality	Major type 3
15	RESIDENT_CITY	resident_city	Major type 3
16	RESIDENT_STATE	resident_state	Major type 3
17	RESIDENT_POSTAL_CODE	resident_postal_code	Major type 3
18	BIOMETRIC_TEMPLATE_XX	biometric_template_xx	Major type 2
19	NAME_NAT_CHAR	name_nat_char	Major type 3
20	MGMT_NEXT_UPDATE	mgmt_nextupdate	Tag value 0 of major type 6
21	FAMILY_NAME	family_name	Major type 3
22	GIVEN_NAME	given_name	Tag value 0 of major type 6
23	BIRTHDATE	birthdate	Tag value 0 of major type 6
24	ISSUE_DATE	issue_date	Tag value 0 of major type 6
25	EXPIRY_DATE	expiry_date	Major type 3
26	ISSUING_COUNTRY	issuing_country	Major type 3

Item #	Identifier	Param Name	CBOR Type
27	ISSUING_AUTHORITY	issuing_authority	Major type 3
28	DOCUMENT_NUMBER	document_number	Major type 3
29	PORTRAIT	portrait	Major type 2
30	MGMT_LAST_UPDATE	mgmt_lastupdate	Tag value 0 of major type 6
31	MGMT_VALIDITY	mgmt_validity	Tag value 0 of major type 6
32	ONLINE_TOKEN_XXXX	online_token_xxxx	Major type 3
34	DRIVING_PRIVILEGES	driving_privileges	Major type 4
35	REAL_ID	RealID	Major type 3 //org.aamva namespace
36	SIGNATURE_USUAL_MARK	signature_usual_mark	Major type 3
37	ONLINE_URL_XXXX	online_url_xxxx	Major type 3
38	BIRTHDATE	birth_date	Major type 3
39	BIRTH_PLACE	birth_place	Major type 3
40	NAME_NAT_CHAR	name_national_character	Major type 3

Expected Results

Data Fields	Data type/Unit	Sample Value
administrative_number	Text/String	"1234453"
gender	Text/String	"M"
height	Integer/Centimeter	"157"
weight	Integer/Kg	"56"
eye_color	Text/String	"BLU"
hair_color	Text/String	"BLACK"
birthplace	Text/String	"NY"
resident_address	Text/String	"123, ABC Street"
portrait_capture_date	DateTime/RFC3339	"1985-04-12T23:20:50Z"
age_in_years	Integer/String	"34"
age_birth_year	Integer/String	"1978"
age_over_NN	Boolean/String	"true"
issuing_jurisdiction	Text/String	"NY"
nationality	Text/String	"US"
resident_city	Text/String	"NY"
resident_state	Text/String	"NY"
resident_postal_code	Text/String	"58773"
name_nat_char	Text/String	"US"
mgmt_nextupdate	DateTime/RFC3339	"1985-04-12T23:20:50Z"
family_name	Text/String	"Sample"
given_name	Text/String	"Joe"
birthdate	DateTime/RFC3339	"1985-04-12T23:20:50Z"
issue_date	DateTime/RFC3339	"1985-04-12T23:20:50Z"
expiry_date	DateTime/RFC3339	"1985-04-12T23:20:50Z"
issuing_country	Text/String	"US"
issuing_authority	Text/String	"NY"
document_number	Int/String	"782593823"
portrait	ByteArray/Bytes	
mgmt_lastupdate	DateTime/RFC3339	"1985-04-12T23:20:50Z"
mgmt_validity	DateTime/RFC3339	"1985-04-12T23:20:50Z"

Data Fields	Data type/Unit	Sample Value
driving_privileges	Text/String	"C"
RealID	Boolean/String	true
signature_usual_mark	ByteArray/Bytes	
birth_date	DateTime/RFC3339	"1985-04-12T23:20:50Z"
birth_place	Text/String	"NY"
name_national_character	Text/String	"US"

Expected Results (PDF417 scan)

Data Fields	Data type/Unit	Sample Value
gender	Text/String	"M"
height	Integer/inches	"71"
eye_color	Text/String	"BLU"
resident_address	Text/String	"123, ABC Street"
age_in_years	Integer/String	"34"
age_over_NN	Boolean/String	"true"
resident_city	Text/String	"NY"
resident_state	Text/String	"NY"
resident_postal_code	Text/String	"58773"
family_name	Text/String	"Sample"
birthdate	DateTime/RFC3339	"1985-04-12T23:20:50Z"
issue_date	DateTime/RFC3339	"1985-04-12T23:20:50Z"
expiry_date	DateTime/RFC3339	"1985-04-12T23:20:50Z"
issuing_country	Text/String	"USA"
document_number	Int/String	"782593823"
driving_privileges	Text/String	"C"

Expected Results (Optical Inspection)

Data Fields	Data type/Unit	Sample Value
gender	Text/String	"M"
family_name	Text/String	"Sample"
given_name	Text/String	"Joe"
birthdate	DateTime/RFC3339	"1985-04-12T23:20:50Z"
expiry_date	DateTime/RFC3339	"1985-04-12T23:20:50Z"

Error Codes

Error Code	Error Code Message	Description	Action Required
101	Invalid QR Code	Unable to scan the QR code.	Please try again.
102	QR Data item is missing or found any issue	QR Data Issue.	Unable to find some elements in the QR data item. The **Mobile ID** reader may abort the transaction.
103	QR Code parsing error	Unable to parse the **Mobile ID App** QR code.	Unable to parse the QR code that the mID has provided.
104	QR code scan error	Unable to Scan QR code may be because of invalid data or bad encoding.	Unable to scan the QR code.
105	Error PDF417 decoder	Error in generating PDF417 barcode	
106	Parsing Error	Error in compact mode qr code parsing and verification	
107	Image Conversion Error	Error in convert image from bitmap to Image	
108	License Validation for Optical Inspection Failed	Please try again with stable internet connection	Try Again.
109	License expired	Please contact support to renew the license	
121	Error in security verification	VerifyID could not validate one or more security checks on response received from the **IDEMIA Mobile ID App**.	
122	Generic Error	Generic Error from VerifyID abstraction layer. Concrete class must override to throw actual error.	

Error Code	Error Code Message	Description	Action Required
201	Unable to initialize BLE	Error in initializing BLE feature of this device.	The **Mobile ID** reader is not able to get BLE. The **Mobile ID** reader may abort the transaction.
202	Device Doesn't Support BLE	This device doesn't support the BLE feature.	The **Mobile ID** reader is not able to get BLE. The **Mobile ID** reader may abort the transaction.
203	BLE Scanning Error	Scanning of the BLE devices failed.	The **Mobile ID** reader is not able to scan the BLE devices. The **Mobile ID** reader may abort the transaction.
204	Bluetooth is OFF	This device's Bluetooth is off.	The Bluetooth of the device is not enabled. The **Mobile ID** reader may abort the transaction.
205	BLE Scanning Error	Please provide the `BLUETOOTH` permissions	
206	BLE Scanning Error	Please enable Bluetooth to get scan results	
207	BLE Scanning Error	Please provide the `BLUETOOTH_ADMIN` permissions	
208	BLE Connection Timeout	Unable to create BLE communication channel with the **Mobile ID** device.	BLE connection failure. The **Mobile ID** reader may abort the transaction.

Error Code	Error Code Message	Description	Action Required
209	BLE Scanning Error	Please provide `ACCESS_COARSE_LOCATION` or `ACCESS_FINE_LOCATION` permission in manifest to get scan results	
210	BLE Scanning Error	Please provide `ACCESS_COARSE_LOCATION` or `ACCESS_FINE_LOCATION` permission at runtime to get scan results	
211	Connection mode not available	**IDEMIA Verify SDK - Android** is not able to setup connection with the **Mobile ID App** back-end using the configured connection type/mode.	
301	Connection Lost	Error occurred while transferring the data.	The **Mobile ID** reader is not able to maintain the connection with the device. The **Mobile ID** reader may abort the transaction.
302	Error Creating Request	Unable to create the request for the mID credential.	The **Mobile ID** reader is not able to create the request. The **Mobile ID** reader may abort the transaction.
303	Transaction Cancelled	The transaction has been cancelled by the end-user.	Unable to scan the QR code.
304	Connection Failed	Unable to connect the device.	There is a BLE connection failure. The **Mobile ID** reader may abort the transaction.
305	Device Disconnected	The device has disconnected. Maybe the device is not in the range of BLE.	The **Mobile ID** reader may abort the transaction.

Error Code	Error Code Message	Description	Action Required
306	Timed Out, Please try again		BLE connection failure. The **Mobile ID** reader may abort the transaction.
307	Request Incomplete	Please select one or more attributes to be included in the request to the **Mobile ID App**.	
308	Incomplete Request	A **Mobile ID App** request comes to an end of data when expecting more data. For example, a **Mobile ID** credential expects a certain length, number of arrays elements, or map entries but instead encounters the end of the data. The **Mobile ID App** back-end does not return any data but the error message	The **Mobile ID** reader may inspect the request, ensure request is complete, and resend it. The **Mobile ID** reader may abort the transaction.
309	Request Rejected	The **Mobile ID** back-end indicates that the request is rejected.	
310	**Mobile ID** reader authentication error	The **Mobile ID** back-end indicates there is an error with **Mobile ID** reader authentication.	
311	General Error	The **Mobile ID** back-end returns an error without any given reason.	The **Mobile ID** reader may inspect the request, ensure request is complete, and resend it. The **Mobile ID** reader may abort the transaction.
400	Error in response data received	Response data is malformed or `nil`	

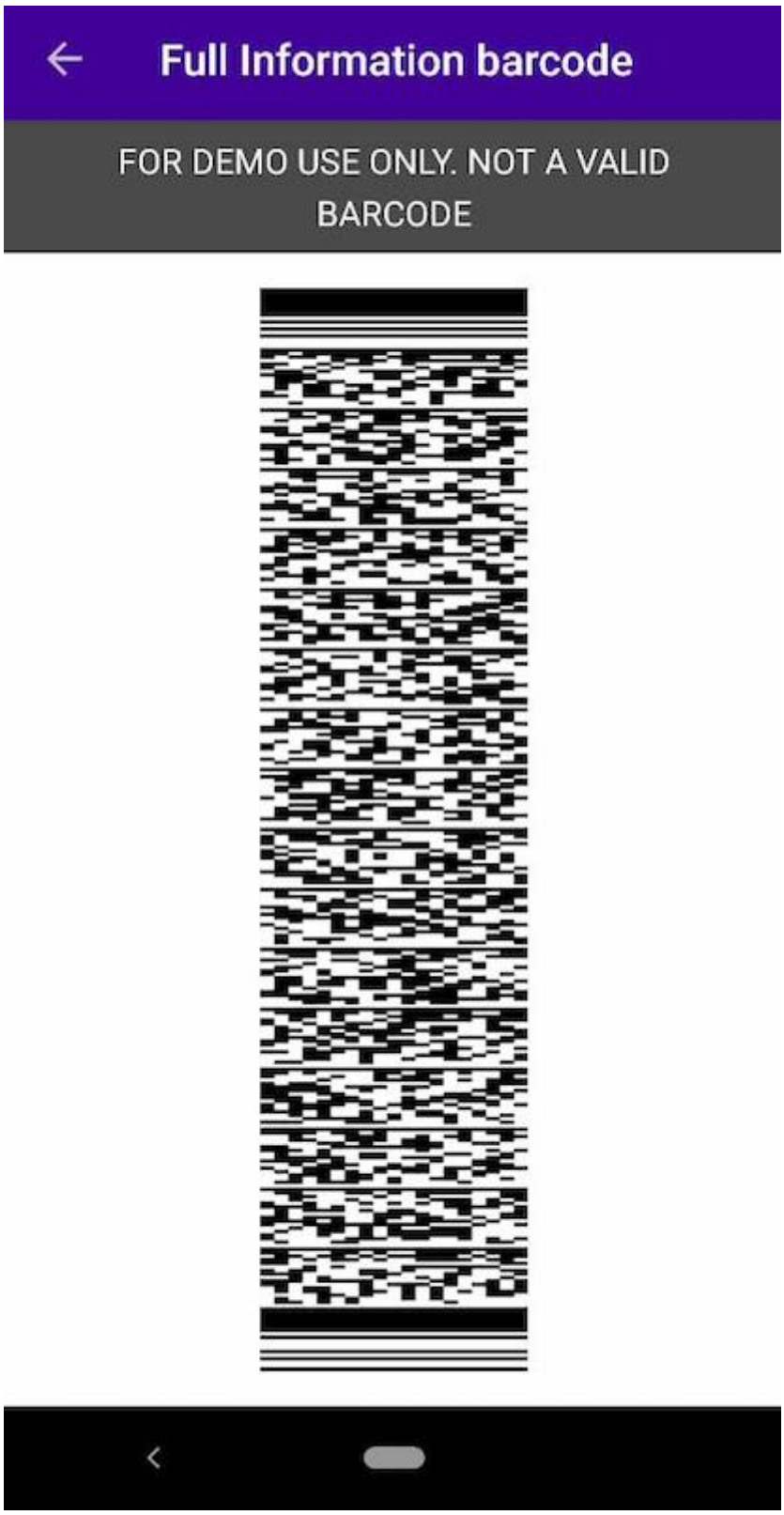
Error Code	Error Code Message	Description	Action Required
401	Parsing Error	A **Mobile ID** credential request encountered a data element that is not well-formed (e.g., invalid initial=byte) and failed decoding the data. The **Mobile ID App** back-end does not return any data but the error message.	The **Mobile ID** reader may inspect the request, ensure request is complete, and resend it. The **Mobile ID** reader may abort the transaction.
402	Invalid Format	The **Mobile ID** back-end cannot process the requested data element due to a formatting error.	
403	Data Not Found	The requested NameSpace or data element within a NameSpace is not found.	
404	Data Request Denied	The release of requested data element was rejected by the **Mobile ID** credential holder.	
405	Data Not Returned	The **Mobile ID** back-end does not provide the requested data element without any given reason.	

Test Samples

Line code Sample

Optical Inspection Sample

PDF417 Sample



QR + BLE Sample

Note: The below screenshot is only for device engagement QR code testing. The full QR + BLE scenario will only work with a real device with the **IDEMIA Mobile ID App** installed.


9:41
Share ID Information
SAMPLE
Place this code in front of the reader to initiate transaction.