Software Development Department

BioSelfie

AAR Implementation Guide

Prepared by
Elia Yakoumi
Version # 1.0.0 Updated on March 6, 2023

Acknowledgments

The contribution of the following individuals in preparing this document is gratefully acknowledged:

[**Contributors/reviewers/developers**]

Role	Name	Phone #	E-Mail Address
Owner	Elia Yakoumi		elia.yakoumi@orbisholding.com
Author	Elia Yakoumi		elia.yakoumi@orbisholding.com
Contributor			
Reviewer	Ibrahim Bizri		Ibrahim.bizri@orbisholding.com
Approval	Antoine Chamieh		Antoine.chamieh@orbisholding.com

Document Number	1.0.0
Document Name	BioSelfie AAR Implementation Guide
Date Created (Draft)	6/3/2023
Date Approved	-/-/2023
Medium of Distribution	Electronic
Security Classification	Private

Information Exchange Protocol:

Version Control

Version	Date	Author	Change Description
1.0.0	06/03/2023	Elia Yakoumi	Document created

Table of Contents

1 OBJECTIVES

The purpose of the current document is describing the mobile SDK integration, in the context of Android environment with the main mobile application.

2 SDK INTEGRATION

2.1 AAR Description

Orbis will be providing 2 AARS for the BioSelfie SDK. One AAR will be for the android devices supporting Google Mobile Services, the other will be for devices with no Google support (Huawei).

Also 2 extra AARs will be provided with the SDK that will be needed for scanning the Emirates ID chip process.

2.2 Required Permissions

To run the BioSelfie mobile SDK the following permissions are required:

a. Camera
b. Record audio
c. Internet
d. NFC

NFC is required for scanning the e-passport and e-ID chips. It is the duty of the calling app to confirm that the device has NFC and it is enabled.

2.3 Gradle Dependencies:

For the SDK to function, external dependencies are required to be added to the gradle file. The common dependencies:

Code:

```
def camerax_version = "1.0.0-beta05"
implementation "androidx.camera:camera-core:${camerax_version}"
implementation "androidx.camera:camera-camera2:${camerax_version}"
implementation "androidx.camera:camera-lifecycle:${camerax_version}"
implementation"androidx.camera:camera-view:1.0.0-alpha12"
implementation "androidx.camera:camera-extensions:1.0.0-alpha12"

implementation 'com.google.android.gms:play-services-mlkit-text-recognition:18.0.2'
implementation 'pl.droidsonroids.gif:android-gif-drawable:1.2.25'
implementation 'com.madgag.spongycastle:prov:1.54.0.0'
implementation 'com.madgag.spongycastle:core:1.54.0.0'
implementation 'com.madgag.spongycastle:pkix:1.54.0.0'
implementation 'com.github.lzyzsd:circleprogress:1.2.1'
implementation 'org.jmrtd:jmrtd:0.7.34'
implementation 'net.sf.scuba:scuba-sc-android:0.0.23'
implementation 'androidx.exifinterface:exifinterface:1.3.5'
implementation 'com.github.Tgo1014:JP2ForAndroid:1.0.4'
implementation("com.vanniktech:android-image-cropper:4.4.0")
```

Also adding the following AARs in the gradle file (depend on the location of the files)

Code:

```
implementation files('../BioSelfie/BioSelfie.aar')
implementation files('../EIDAToolkit/EIDAToolkit.aar')
implementation files('../NFC/NFC.aar')
```

Add the following maven repo to the project (settings.gradle file)
Code:

```
maven { url 'https://jitpack.io' }
```

For google devices this dependency is required:
Code:

```
implementation 'com.google.android.gms:play-services-mlkit-text-recognition:18.0.2'
```

For non Google devices
Code:

```
implementation 'com.huawei.hms:ml-computer-vision-ocr:3.5.0.304'
```

2.4 Minimum SDK:

The minimum android version to run BioSelfie is android 7.1 (SDK 25)

3 FUNCTIONALITIES:

The BioSelfie SDK functionalities are called using intents. For each of the below described functionality, a start activity for result is used to start the intent and await the result

Before calling any scanning activity, a function to reset data is called:
Code:

```
BioKYCLib.resetData();
```

Before calling any Matching activity, a function to reset Biometric data is called:
Code:

```
BioKYCLib.resetMatchingData();
```

3.1 ID Scanning:

ID Scanning is done using this functionality. Depending on the input parameters, the scan will be for front, back or double side of the ID.
After the Scan of the ID the system will determine the type of the ID automatically if it is a Qatari ID (QID), an MRID, an e-ID (French ID) or an Emirates ID and returns the data correspondingly.

Calling app have the liberty to scan each side of the ID by itself or do the full process by scanning both sides of the ID.

In case the calling app decides to scan the sides independently, on the scan of the back side of the ID, the issuing country (that will be returned after scanning the front side of the ID) should be passed to the function to proceed.

In case of an e-ID (French ID), a flag in the OpticalResult class will be set to true. In this case the calling app can start reading the NFC chip of this e-ID by calling the scanning functionality

Also, in case of an Emirates ID, a flag OpticalResult class will be set to true. In this case the calling app can start reading the NFC chip of the emirates ID by calling the scanning functionality.

The related Result objects for the ID scanning:

a- Results.Result
b- Results.OpticalResult
c- Results.ElectronicResult (in case of required scanning of Emirates and e-IDs).

Description of these results are described in the Result section of this document.

To start the scan an intent to start the activity "**IDScanMain**" is to be used

When calling the scan functionality, some inputs are to be provided to the calling intent as follows:

Extra Name	Type	Description	Discussion
url	String	The url where the services are installed	

SCAN_CAPTION	String	Title displayed on the scan screen	In case not passed, a hardcoded default value will be used
SHOW_LOGO	Boolean	Display BioSelfie logo during scanning	- In case not passed, default value is false - Pressing the logo will redirect to the BioSelfie Web Page
CLIENT_NAME	String	client name used in BioSelfie license	
SCAN_FLOW	String	flow for scanning	Can be one of the following: - "front": to scan front of the ID - "back": to Scan back of the ID. In this case a country should be specified as input (refer COUNTRY below). - "double" to scan both sides of the ID.
SHOW_GUIDE	Boolean	display scanning guidelines	In case not passed, default value is false
COUNTRY	String	ID issuing country	in case the flow is to scan back side of ID this is a mandatory field to be passed
IS_IMAGE_BUTTONS	Boolean	Whether the proceed/reupload buttons are normal buttons or image buttons	
PROCEED_IMAGE	String	the path for the proceed button image	Refer notes 1, 2
RETAKE_IMAGE	String	the path for the retake button image	Refer notes 1, 2
RETAKE_CAPTION	String	Text shown on the retake button	Refer notes 3, 4
PROCEED_CAPTION	String	Text shown on the proceed Button	Refer notes 3, 4

Note 1: SDK will check these inputs only in case IS_IMAGE_BUTTONS is set to true otherwise these 2 inputs can be ignored.
Note 2: If paths are not supplied while **using image buttons,** default images shipped with the aar will be used as default
Note 3: SDK will check these inputs only in case IS_IMAGE_BUTTONS is set to false otherwise these 2 inputs can be ignored.
Note 4: If texts are not supplied while **not using image buttons,** default texts will be used as default

A sample call for the scan Intent:

Code:

```
public void start(View view) {
        BioSelfieLib.resetData();
        Intent intent = new Intent(this, IDScanMain.class);
        intent.putExtra("url", urlEntry.getText().toString().trim());
        intent.putExtra("SCAN_CAPTION", "Scan ID");
        intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
         intent.putExtra("SCAN_FLOW", type);
        if (type.equalsIgnoreCase("back"))
                intent.putExtra("COUNTRY", ctrEntry.getText().toString().trim());
        intent.putExtra("SHOW_LOGO", true);
        intent.putExtra("IS_IMAGE_BUTTONS", true);
        intent.putExtra("SHOW_GUIDE", true);
        startActivityForResult(intent, REQUEST);
}
```

The result of the call can be handled with on Activity result function

Sample of result handling function:

Code:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)
{
  super.onActivityResult(requestCode, resultCode, data);
  if (requestCode == REQUEST)
  {
        if (resultCode == RESULT_CANCELED)
        {
                Toast.makeText(this, "Result Code: " + Result.errorCode + " Result
        Message: "+ Result.message, Toast.LENGTH_LONG).show();
        }
        else
        {
                Intent intent = new Intent(this, Results.class);
                intent.putExtra("url", urlEntry.getText().toString().trim());
                intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
                startActivity(intent);
        }
  }
}
```

After doing the OCR calling app can check the type of the ID and start scanning the corresponding chip by calling the EID or emirates id scanning functionality

To start the chip scanning functionality for an **EID** the following inputs are required:

Extra Name	Type	Description	Discussion
url	String	The url where the services are installed	
CLIENT_NAME	String	client name used in BioSelfie license	

To start the scan of an EID chip an intent to start the activity "**ScanEID**" is to be used

A sample call:
Code:

```
Intent intent = new Intent(this, ScanEID.class);
intent.putExtra("url", urlEntry.getText().toString().trim());
intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
startActivityForResult(intent, 9001);
```

To start the scan of an Emirates ID chip an intent to start the activity "**StartEmiratesNFCScan**" is to be used

A sample call:
Code:

```
Intent intent = new Intent(this, StartEmiratesNFCScan.class);
startActivityForResult(intent, 9001);
```

The result of the call can be handled with on Activity result function

3.2 ID Upload:

Scanning an ID from the device gallery is done using this functionality. Depending on the input parameters, the upload and ocr will be for front, back or double side of the ID.

Calling app have the liberty to scan each side of the ID by itself or do the full process by scanning both sides of the ID.

In case the calling app decides to scan the sides independently, on the scan of the back side of the ID, the issuing country (that will be returned after scanning the front side of the ID) should be passed to the function to proceed.

The related Result objects for the ID scanning:
a- Results.Result
b- Results.OpticalResult
Description of these results are described in the Result section of this document.

To start the scan an intent to start the activity "**IDUploadMain**" is to be used

When calling the scan functionality, some inputs are to be provided to the calling intent as follows:

Extra Name	Type	Description	Discussion
url	String	The url where the services are installed	
SCAN_CAPTION	String	Title displayed on the scan screen	In case not passed, a hardcoded default value will be used
SHOW_LOGO	Boolean	Display BioSelfie logo during scanning	- In case not passed, default value is false - Pressing the logo will redirect to the BioSelfie Web Page
CLIENT_NAME	String	client name used in BioSelfie license	
SCAN_FLOW	String	flow for scanning	Can be one of the following: - "front": to scan front of the ID - "back": to Scan back of the ID. In this case a country should be specified as input (refer COUNTRY below). - "double" to scan both sides of the ID.
SHOW_GUIDE	Boolean	display scanning guidelines	In case not passed, default value is false
COUNTRY	String	ID issuing country	in case the flow is to scan back side of ID this is a mandatory field to be passed
IS_IMAGE_BUTTONS	Boolean	Whether the proceed/reupload buttons are normal buttons or image buttons	
PROCEED_IMAGE	String	the path for the proceed button image	Refer notes 1, 2

RETAKE_IMAGE	String	the path for the retake button image	Refer notes 1, 2
RETAKE_CAPTION	String	Text shown on the retake button	Refer notes 3, 4
PROCEED_CAPTION	String	Text shown on the proceed Button	Refer notes 3, 4

Note 1: SDK will check these inputs only in case IS_IMAGE_BUTTONS is set to true otherwise these 2 inputs can be ignored.
Note 2: If paths are not supplied while **using image buttons,** default images shipped with the aar will be used as default
Note 3: SDK will check these inputs only in case IS_IMAGE_BUTTONS is set to false otherwise these 2 inputs can be ignored.
Note 4: If texts are not supplied while **not using image buttons,** default texts will be used as default

A sample call for the scan Intent:
Code:

```
public void upload(View view) {
    BioSelfieLib.resetData();
    Intent intent = new Intent(this, IDUploadMain.class);
    intent.putExtra("url", urlEntry.getText().toString().trim());
    intent.putExtra("SCAN_CAPTION", "Upload ID");
    intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
    intent.putExtra("SCAN_FLOW", type);
    if (type.equalsIgnoreCase("back"))
        intent.putExtra("COUNTRY", ctrEntry.getText().toString().trim());
    intent.putExtra("SHOW_LOGO", true);
    intent.putExtra("IS_IMAGE_BUTTONS", true);

    startActivityForResult(intent, REQUEST);
}
```

The result of the call can be handled with on Activity result function

Sample of result handling function:

Code:

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST)
        {
            if (resultCode == RESULT_CANCELED)
            {
                Toast.makeText(this, "Result Code: " + Result.errorCode + " Result Message: "+
Result.message, Toast.LENGTH_LONG).show();
            }
            else
            {
                    Intent intent = new Intent(this, Results.class);
                    intent.putExtra("url", urlEntry.getText().toString().trim());
                    intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
                    startActivity(intent);
                }
        }
    }
```

3.3 Passport Scanning:

Passport Scanning is done using this functionality. Depending on the input parameters, this will be a normal passport scan, a forced e-passport scan or an autodetect passport scan.

In case of an e-passport scan, after the passport has been OCRed correctly, a NFC chip scan is started to read the details of the chip. This step cannot be bypassed otherwise an error will be returned

In case of an auto scan, a flag is set to tell calling app that the scanned passport is an e-passport or not and decide accordingly to start an NFC scan. The user can cancel the scan for the NFC and the optical result is only returned.

The related Result objects for the ID scanning:

a- Results.Result
b- Results.OpticalResult
c- Results.ElectronicResult (in case of required scanning of e-ppassports).

Description of these results are described in the Result section of this document.

To start the scan an intent to start the activity "**PassCamera**" is to be used

When calling the scan functionality, some inputs are to be provided to the calling intent as follows:

Extra Name	Type	Description	Discussion
url	String	The url where the services are installed	
SCAN_CAPTION	String	Title displayed on the scan screen	In case not passed, a hardcoded default value will be used
SHOW_LOGO	Boolean	Display BioSelfie logo during scanning	- In case not passed, default value is false - Pressing the logo will redirect to the BioSelfie Web Page
CLIENT_NAME	String	client name used in BioSelfie license	
SCANNING_TYPE	String	type of passport to scan	can be one of the following: - "auto" for passport autodetection - "Optical" for non biometric passports - "Electronic" for forcing e-passport scan In case of auto scan, user can bypass chip scanning

A sample call for the scan Intent:

Code:

```
public void start(View view) {
    if (!urlEntry.getText().toString().trim().equalsIgnoreCase("") &&
            !clientEntry.getText().toString().trim().equalsIgnoreCase(""))
    {
        BioSelfieLib.resetData();
        Intent intent = new Intent(this, PassCamera.class);
        intent.putExtra("url", urlEntry.getText().toString().trim());
        intent.putExtra("SCAN_CAPTION", "Scan Passport");
        intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
        intent.putExtra("SCANNING_TYPE", type);
        intent.putExtra("SHOW_LOGO", true);
        intent.putExtra("SHOW_GUIDE", !cb1.isChecked());
        startActivityForResult(intent, REQUEST);

    }
}
```

The result of the call can be handled with on Activity result function

Sample of result handling function:
Code:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST)
    {
        if (resultCode == RESULT_CANCELED)
        {
            Toast.makeText(this, "Result Code: " + Result.errorCode + " Result Message: "+
Result.message, Toast.LENGTH_LONG).show();
        }
        else
        {
            Intent intent = new Intent(this, Results.class);
            intent.putExtra("url", urlEntry.getText().toString().trim());
            intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
            startActivity(intent);
        }
    }
}
```

In case of Auto passport detection is chosen and the passport is detected as an e-passport by checking the flag isEpass from the Optical result class, an NFC scan can be launched.

To start the chip scanning functionality the following inputs are required:

Extra Name	Type	Description	Discussion
url	String	The url where the services are installed	

CLIENT_NAME	String	client name used in BioSelfie license	

To start the scan of the NFC chip an intent to start the activity "**StartNFC**" is to be used

A sample call:
Code:

```
public void scan(View view) {
        Intent intent = new Intent(this, StartNFC.class);
        intent.putExtra("url", url);
        intent.putExtra("CLIENT_NAME", clientName);
        startActivityForResult(intent, CODE);
    }
```

The result of the call can be handled with on Activity result function

3.4 Passport Upload:

This function is used to do OCR for a passport image in the device gallery.

To start the upload, an intent to the activity "**UploadAndClassifyPassport**" is to be created and called for result.

Once Upload and OCR is done, the result of the activity can be either a RESULT_OK specifying the success of the passport reading or a RESULT_CANCELLED specifying an error during upload.

The related Result objects for the passport scanning:

a- Results.Result
b- Results.OpticalResult

Description of these results are described in the Result section of this document.

Some input parameters are required for the activity to start. Those are passed as extras to the intent

Extra Name	Type	Description	Discussion
url	String	The url where the services are installed	
SCAN_CAPTION	String	Title displayed on the scan screen	In case not passed, a hardcoded default value will be used
SHOW_LOGO	Boolean	Display BioSelfie logo during scanning	- In case not passed, default value is false - Pressing the logo will redirect to the BioSelfie Web Page
CLIENT_NAME	String	client name used in BioSelfie license	
IS_IMAGE_BUTTONS	Boolean	Whether the proceed/reupload buttons are normal buttons or image buttons	
PROCEED_IMAGE	String	the path for the proceed button image	Refer notes 1, 2
RETAKE_IMAGE	String	the path for the retake button image	Refer notes 1, 2
RETAKE_CAPTION	String	Text shown on the retake button	Refer notes 3, 4
PROCEED_CAPTION	String	Text shown on the proceed Button	Refer notes 3, 4

Note 1: SDK will check these inputs only in case IS_IMAGE_BUTTONS is set to true otherwise these 2 inputs can be ignored.
Note 2: If paths are not supplied while **using image buttons,** default images shipped with the aar will be used as default
Note 3: SDK will check these inputs only in case IS_IMAGE_BUTTONS is set to false otherwise these 2 inputs can be ignored.
Note 4: If texts are not supplied while **not using image buttons,** default texts will be used as default

A sample call for the Passport Upload

Code:

```
public void upload(View view) {
        BioSelfieLib.resetData();
        Intent intent = new Intent(this, UploadAndClassifyPassport.class);
        intent.putExtra("url", urlEntry.getText().toString().trim());
        intent.putExtra("SCAN_CAPTION", "Upload Passport");
        intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
        intent.putExtra("SHOW_LOGO", true);
        intent.putExtra("IS_IMAGE_BUTTONS", true);
        startActivityForResult(intent, REQUEST);
    }
```

Sample for result handling:

Code:

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST)
        {
            if (resultCode == RESULT_CANCELED)
            {
                Toast.makeText(this, "Result Code: " + Result.errorCode + " Result Message: "+
Result.message, Toast.LENGTH_LONG).show();
            }
            else
            {
                Intent intent = new Intent(this, Results.class);
                intent.putExtra("url", urlEntry.getText().toString().trim());
                intent.putExtra("CLIENT_NAME", clientEntry.getText().toString().trim());
                startActivity(intent);

            }
        }
}
```

3.5 Match Image to Selfie (Video or Still Image):

After OCR is done (and NFC reading if applicable), matching between the document face image and a video/ Image of the person face is to be done.
To match, first we need to set the face image. To do so we need to call the following:
BioSelfieLib.setMatchingImage(Bitmap image);
where image is the face image derived from the optical or electronic result.

To start matching, an intent to the matching activity "**TakeMatchingVideo**" is to be created and called for result.
If ICAO compliance is required, the intent to be called "**TakeMatchingICAOVideo**"

Once OCR is done, the result of the activity can be either a RESULT_OK specifying the success of the passport reading or a RESULT_CANCELLED specifying an error during OCR.

In case of enrollment, an ID is generated during ocr. But this id can also be set by calling app by calling the function:
BioSelfieLib.setID(String type, String number, String DOB, String DOE)

Where:
a- type can be "P" for passport, "QID" for "QID" or ID for other IDs
b- number is the Document number (or QID number)
c- DOB: date of birth (can be empty for QID)
d- DOE: expiry date (can be empty for QID)

The related Result objects for the passport scanning:

a- Results.Result
b- Results.BiometricResult

Description of these results are described in the Result section of this document.

Some input parameters are required for the activity to start. Those are passed as extras to the intent

Extra Name	Type	Description	Discussion
url	String	The url where the services are installed	
SCAN_ SELFIE_CAPTION	String	Title displayed on the camera screen	In case not passed, a hardcoded default value will be used
isImage	Boolean	Specify whether capture an image or a video	
forceEnroll	Boolean	In case Enrollment is required, force enroll is used to enroll if user matches another ID or the current image doesn't match existing images	
doEnroll	Boolean	Enroll user after matching	
CHECK_LIVE	Boolean	Whether do liveness check or not	
SHOW_LOGO	Boolean	Display BioSelfie logo during scanning	- In case not passed, default value is false - Pressing the logo will redirect to the BioSelfie Web Page

CLIENT_NAME	String	client name used in BioSelfie license	
SHOW_GUIDE	Boolean	display guidelines	In case not passed, default value is false.

Note that liveness test is NOT available for image to image matching

A sample call for the Matching:
Code:

```
public void matchImageVideo(View view) {
    BioSelfieLib.resetMatchingData();
    if (ElectronicResult.faceImage!= null)
       BioSelfieLib.setMatchingImage(ElectronicResult.faceImage);
    else
       BioSelfieLib.setMatchingImage(OpticalResult.faceImage);
    boolean isICAO = cb1.isChecked();
    Intent intent;
    if (!isICAO)
       intent = new Intent(this, TakeMatchingVideo.class);
    else
       intent = new Intent(this, TakeMatchingICAOVideo.class);
    intent.putExtra("url", url);
    intent.putExtra("SELFIE_CAPTION", "Capture Selfie");
    intent.putExtra("isImage", false);
    intent.putExtra("doEnroll", false);
    intent.putExtra("CLIENT_NAME", clientName);
    if(cb3.isChecked())
       intent.putExtra("SHOW_LOGO", true);
    if(cb4.isChecked())
       intent.putExtra("CHECK_LIVE", true);
    else
       intent.putExtra("CHECK_LIVE", false);
    intent.putExtra("SHOW_GUIDE", !cb5.isChecked());
    startActivityForResult(intent, 1001);
  }
```

A sample for result handling:
Code:

```
@Override
  protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    Intent intent = new Intent(this, BioMetricActivity.class);
    startActivity(intent);
  }
```

4 RESULTS:

4.1 Optical Result:

This class holds the data derived by doing the OCR on the provided Document.
(Please note that not all fields are applicable/Found for the ID)

Name	Type	Value
documentType	String	Document Type (ID for Qatari ID, or passport type)
dateOfExpiry	Date	Date Of Expiry of the Document In Case of QID and the Expiry date was not available, today's date is returned
nationality	String	Nationality
documentNumber	String	The document Number
givenNames	String	In case of passport the given name In case of ID the full name
surname	String	Family name in case of passport
dateofBirth	Date	Date Of Birth In case of passport format is yymmdd In case of ID format is dd/mm/yyyy
faceImage	Bitmap	Bitmap image containing the face on the document
faceImageB64	String	Base64 string representation of the face image
documentImage	Bitmap	The Document image
documentImageB64	String	Base64 string representation of the document image
documentImageBack	Bitmap	Back side document image
documentImageB64Back	String	Base64 string representation of the document back side image
gender	String	Gender
issuingState	String	Issuing state of the document in case of passport
personalNumber	String	Represent the personal number on a passport if present
classificationScore	double	When using classification this represents the score of the classification
forgeryScore	double	Probability of a forged document
classificationCountry	String	Country of which this document is classified for
optionalFiled1	String	On MRID this represents the optional field 1
optionalField2	String	On MRID this represents the optional field 2
isEpass	Boolean	States whether the scanned passport is an epassport
isEID	Boolean	States whether the scanned ID is an electronic ID (French) or not
isEmirates	Boolean	States whether the scanned ID is an emirates id or not
qidPassportExpiryDate	Date	Passport expiry date on the back side of the QID
qidPassportNumber	String	Passport number on the back side of the QID

4.2 Electronic Result:

This class holds the data extracted from the **E-passport** chip only. For the optical data from the e-passport refer to optical results

Name	Type	Value
documentType	String	Document Type (ID for Qatari ID, or passport type)
dateOfExpiry	Date	Date Of Expiry of the Document with format yymmdd
nationality	String	Nationality
documentNumber	String	The document Number
givenNames	String	given name
surname	String	Family name
dateofBirth	Date	Date Of Birth with format yymmdd
faceImage	Bitmap	Bitmap image containing the face image extracted from the chip
faceImageB64	String	Base64 String representation of the face image extracted from the chip
gender	String	Gender
issuingState	String	Issuing state of the document
chipCheckResult	String	Chip Verification Result
personalNumber	String	Represent the personal number on a passport if present
optionalField1	String	MRID optional field 1
optionalField2	String	MRID optional field 2
emiratesData	CardPublicData	Emirates ID Data

4.3 Result:

errorCode: integer represent status of the process
message: string small description of the error
the error codes and messages are displayed in the below table

Value	Message
0	Matching done successfully
1	Document scanned successfully
-1	The operation was canceled by the user
-2	Network Error
-3	Scan timeout
-4	Error while reading the provided document
-5	NFC Failed
-6	License Expired
-7	Unable to open the camera
-8	Initialization Error
-9	message sent from the WS
-10	NFC Not Available
-11	Document classification score + classificationScore + is too low to proceed

4.4 BiometricResult:

Name	Type	Value
isLive	Boolean	describe liveness status of the provided video. In case false, the errorCode inside Result class will be 1 with the message variable displaying the not live message.
MatchScore	double	matching score between provided image and video. This value should be compared to the threshold on the application side to decide whether it is a match or not
EnrolledImage	Bitmap	image used to enroll after matching. (in case of image matching it is the image captured, in case of a video matching it is a frame selected from the video)
EnrollmentMessage	String	outcome of the enrollment
CapturedImage	Bitmap	Captured image used for matching.
eye_distance_compliant	Boolean	ICAO eye distance compliance
open_eyes_compliant	Boolean	ICAO eye open compliance
eye_wear_compliant	Boolean	ICAO eye wear compliance
face_clear_compliant	Boolean	ICAO face clear compliance
frontal_pose_compliant	Boolean	ICAO face frontal pose compliance
sufficiant_light_compliant	Boolean	ICAO sufficient light compliance
uniform_light_compliant	Boolean	ICAO uniform lighting compliance
blur_compliant	Boolean	ICAO image blur compliance
mouth_closed_compliant	Boolean	ICAO close mouth compliance
head_position_compliant	Boolean	ICAO head position in image compliance