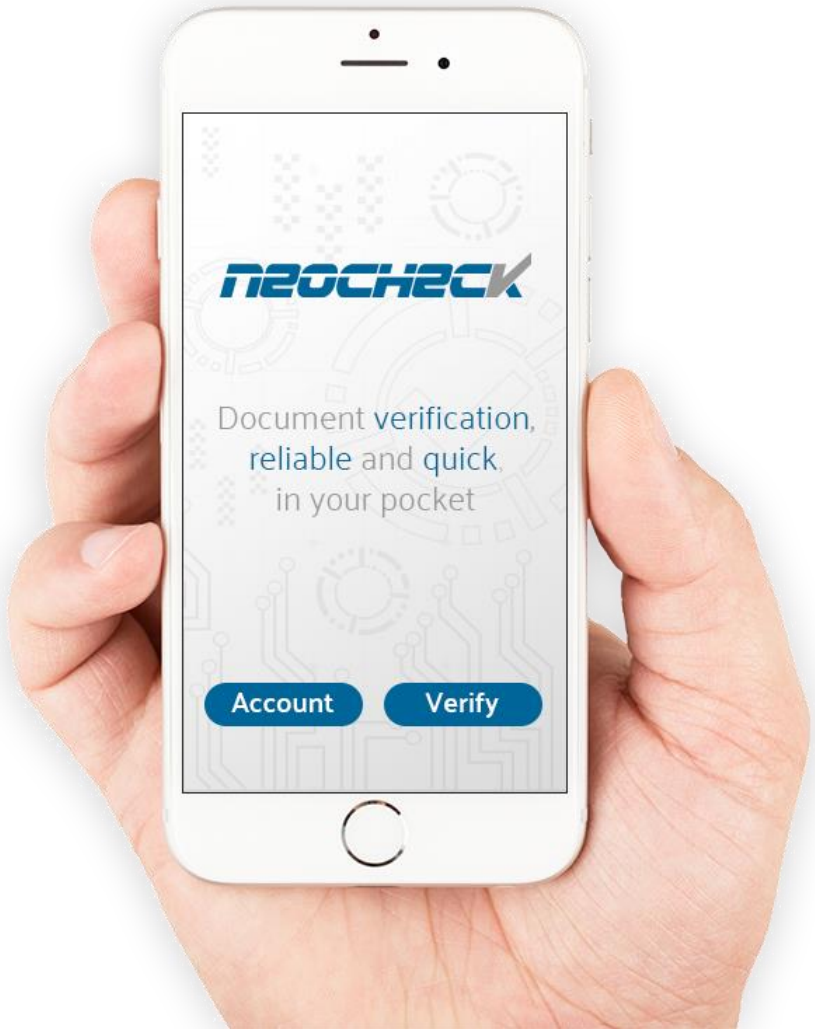


ANDROID SDK

Developers Guide

Content description

This is a reference manual and configuration guide for the NeoCheck Document Verification Android SDK product. It shows how to interact with the Xamarin Android activity to verify all kind of ID and travel security documents in a fast and easy way.

Madrid, May 7th 2017

Disclaimer

Whilst careful attention have been paid to ensure the appropriate referencing of information and sources, it is possible that a document of this nature might contain some errors. In such cases, upon notification NeoCheck will immediately analyse and make any required corrections or amendments.

Release Notes

Version	Date	Description
1.0	07/05/2017	Initial Version

Index

1. Introduction

This user manual describes the different functions and features provided by NeoCheck Android SDK. The purpose of this SDK is to provide on-device fast MRZ OCR and validation from ICAO 9303 compliant documents (passport, visas, ID cards) and ISO 18013 compliant documents (driver's licenses).

Features:

- On-device processing. No need for network connection
- Works with both live video streams and saved images
- High performance with high accuracy
- Validation of dates
- Automatic MRZ detection on the whole image
- Parsing MRZ lines into separate fields
- Non-standard MRZ validation
- Verification of check digits
- Easy to integrate into any application

2. Xamarin Installation

NeoCheck Android SDK was created with Xamarin, a cross-platform implementation of Microsoft .NET. Xamarin products rely upon the platform SDKs from Apple and Google to target iOS or Android.

2.1. Development Environments

This table shows which platforms can be built with different development tool & operating system combinations:

	MACOS	WINDOWS
Development Environment	**VISUAL STUDIO FOR MAC**	**VISUAL STUDIO**
Xamarin.iOS	Yes	Yes (with Mac computer)
Xamarin.Android	Yes	Yes

NOTE: To develop for iOS on Windows computers there must be a Mac computer accessible on the network, for remote compilation and debugging. This also works if you have Visual Studio running inside a Windows VM on a Mac computer.

2.2. macOS Requirements

Using a Mac computer for Xamarin development requires the following software/SDK versions. Check your operating system version and follow the instructions for the Xamarin installer.

	RECOMMENDED	NOTES
Operating System	**OS X El Capitan (10.11) or macOS Sierra**	The minimum required version is OS X El Capitan (10.11).
Xamarin.iOS	**iOS 10 SDK**	This iOS SDK ships with Xcode 8.
Xamarin.Android	**Android 6.0 / API level 23**	You can still target older Android versions while using the latest SDK, or you can build against older versions of the SDK if required.

2.3. Windows Requirements

Using a Windows computer for Xamarin development requires the following software/SDK versions. Check your operating system version (and confirm that you are not using an *Express* version of Visual Studio - if so, consider updating to a *Community* edition). Visual Studio 2015 and 2017 installers include an option to install Xamarin automatically.

	RECOMMENDED	NOTES
Operating System	**Windows 10**	The minimum operating system version is Windows 7. Xamarin.Forms Windows support requires Windows 8.1, and Xamarin.Forms UWP support requires Windows 10.
Xamarin.iOS	**iOS 10 SDK** installed on a Mac	To build iOS projects on Windows requires: Visual Studio 2015 or newer, and a Mac computer, network-accessible from the Windows computer, that conforms to the minimum requirements for running Xamarin on macOS.
Xamarin.Android	**Android 6.0 / API level 23**	You can still target older Android versions while using the latest SDK, or you can build against older versions of the SDK if required.

3. NeoCheck Android SDK

NeoCheck Android SDK comes as a set of .NET libraries, created using Xamarin.Android. Xamarin.Android makes it possible to create native Android applications using the same UI controls as in Java, but using C# language.

3.1. Client application prerequisites

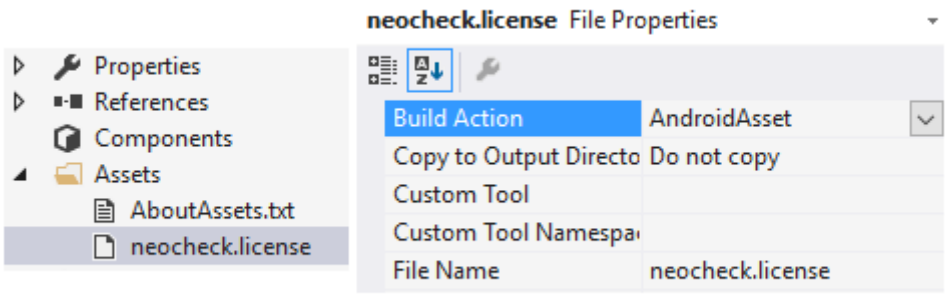
- Android operating system version 5.0 and above
- NuGet package `Xamarin.Android.Support.v4`
- Android permissions:
    - CAMERA
    - FLASHLIGHT

An example of AndroidManifest.xml file for a client application with correct NeoCheck Android SDK configuration is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="NeoCheck.Android.SDK"
android:versionCode="1" android:versionName="1.0" android:installLocation="auto">
        <uses-sdk android:minSdkVersion="21" />
        <uses-permission android:name="android.permission.CAMERA" />
        <uses-feature android:name="android.hardware.camera" android:required="true" />
        <uses-feature android:name="android.hardware.camera.autofocus" />
        <uses-permission android:name="android.permission.FLASHLIGHT" />
</manifest>
```

3.2. Usage

The first step is to copy the NeoCheck Android SDK license to the Assets directory of the Xamarin Android Application that will make use of it (On Visual Studio, once license file is copied to the Assets directory, make sure to set the BuildAction dropdown menu with the value `AndroidAsset`).



3.2.1. Initialization

The first step is to initialize NeoCheck Android SDK, by passing the license file along with the ApplicationContext object. A C# code snippet example is shown below:

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);
    ...
    if (!NeoCheckSDK.Instance.IsInitialized())
        Initialize();
    ...
}
private void Initialize()
{
    try
    {
        byte[] license = new byte[0];
        AssetManager assets = this.Assets;
        using (StreamReader sr = new StreamReader(assets.Open("neocheck.license")))
        {
            using (var memstream = new MemoryStream())
            {
                sr.BaseStream.CopyTo(memstream);
                license = memstream.ToArray();
            }
        }
        var isLicenseValid = NeoCheckSDK.Instance.Initialize(ApplicationContext, license);
    }
    catch (IOException fileNotFoundException)
    {
        //License file not found
    }
    catch (InvalidLicenseException licenseException)
    {
        //Invalid license file
    }
}
```

3.2.2. Automatic MRZ capture and validation

To use the automatic MRZ capture and validation functionality, simply start the ScanActivity provided, and get data captured from NeoCheck Android SDK on activity result. A C# code snippet example is shown below:

```
buttonScan.Click += delegate
{
    try
    {
        Intent intent = new Intent(this, typeof (ScanActivity));
        StartActivityForResult(intent, 0);
    }
    catch (ApplicationException ex)
    {
        //ScanActivity failed to initialize. Make sure license is valid and NeoCheck
        //Android SDK is initialized
    }
};
protected override void OnActivityResult(int requestCode, Result resultCode, Intent data)
{
    base.OnActivityResult(requestCode, resultCode, data);
    if (resultCode == Result.Ok)
    {
        var mrzLines = NeoCheckSDK.Instance.GetMrzLines();
        var mrzImage = NeoCheckSDK.Instance.GetMrzImage();
        var verifications = NeoCheckSDK.Instance.GetVerifications();
        var dataFields = NeoCheckSDK.Instance.GetDataFields();
    }
}
```

3.3. .NET Reference

3.2.3. NeoCheckSDK Class

The main point of entry of the NeoCheck Android SDK is the NeoCheckSDK class. It contains methods to initialize and retrieve capture and verification results. The main properties and methods of this class are detailed below:

```
public static NeoCheckSDK Instance
public IList<string> GetMrzLines()
public byte[] GetMrzImage()
public IList<VerificationItem> GetVerifications()
public IList<DocumentDataField> GetDataFields()
public bool IsInitialized()
```

3.2.4. ScanActivity Android Activity

The `ScanActivity` is an Android Activity that encases all the capture work. It automatically detects MRZ area, takes picture of it when it's focused and performs data extraction and verification. The only step needed is to start the activity and wait for the result. If capture was successful, data and verifications will be available through `NeoCheck` SDK methods, described in the previous chapter.

3.2.5. VerificationItem Class

The `VerificationItem` class represents a verification made over the MRZ, according to ICAO standards, detailed in this document. The main properties of this class are detailed below:

```
public string Key { get; set; }
public VerificationCategory VerificationCategory { get; set; }
public string Name { get; set; }
public VerificationResult VerificationResult { get; set; }
public List<VerificationOutput> VerificationOutputs { get; set; }
public string Output { get; set; }
```

3.2.6. DocumentDataField Class

The `DocumentDataField` class represents a field data extracted from the MRZ, according to ICAO standards. The main properties of this class are detailed below:

```
public string Key { get; set; }
public DocumentSource Source { get; set; }
public string Name { get; set; }
```