MFScan SDK Android

Mantra Softech India Pvt. Ltd.

V 1.0.0.0

MANTRA™

Contents

CONTROL SHEET

Version	1.0.0.0
Release Date	09 August 2024
Author	Mr. Parag Jikadara
Approved By	Mr.Nilesh Prajapati & Mr. Jigar Shekh & Mr. Mahesh Patel
Software Support	servico@mantratec.com 079-49068000 (Extension: 1)

DOCUMENT HISTORY

Document Version	Release Date	Release Notes	Author
1.0.0.0	09/08/2024	Initial release document	Mr.Parag Jikadara

MANTRA™

1. Overview

This document provides the functional and implementation information for working with the below supported devices on Android handsets. By using this SDK, you can capture fingerprints from the below supported devices.

- Please note that you can use only one device at a time with this SDK.

2. Supported Devices

- MFS500
- MARC10
- MELO31

3. Dependency

SDK used for the following dependency is required:

- USB Host Enable Android Handset
- Android OS 5.0 and above
- MFScan.aar with 4 architectures “armeabi-v7a”, “arm64-v8a”, “x86” and “x86_64”

4. Functions

4.1. MFScan Constructor

The MFScan class serves as the primary interface for interacting with the supported device. It provides essential functionalities for device initialization, finger capture, error handling.

JAVA:

Public MFScan (Context context, MFScan_Callback callback);

PARAMETERS:

context= [IN] Application context.

callback= [IN]Registers a callback with overridden methods OnDeviceDetection(), OnPreview(), and OnComplete().

4.2. GetConnectedDevices Function

This method is used to determines the connection status of a specific device.

JAVA:

Public int GetConnectedDevices(List<String>connectedDevices);

PARAMETERS:

connectedDevices = [OUT] list of devices that are compatible with the SDK.

RETURN:

- Returns 0 if connected devices are found and the list is filled.
- Returns an error code otherwise.

4.3. GetSupported Devices Function

This method is used to retrieve the list of devices supported by the SDK.

JAVA:

Public int GetSupportedDevices(List<String>supportedDevices);

PARAMETERS:

supportedDevices = [OUT] A list that will be filled with the SDK with supported devices.

RETURN:

- Returns 0 if supported devices are found.
- Returns an error code otherwise.

4.4. Init Function with clientkey

This method is responsible for initializing the lock device. If USB permission has not been granted, the method first requests the necessary USB permissions before proceeding with the initialization process. The method ensures that only one lock device is initialized at a time, maintaining system integrity and preventing conflicts.

JAVA:

public int Init(final DeviceModel Name , byte[] lockKey ,DeviceInfo deviceInfo)

PARAMETERS:

- **Name:** [IN] Device model name.
- **ClientKey:** [IN] Lock key.
- **deviceInfo:** [OUT] DeviceInfo object for retrieving device information.

RETURN:

- Returns 0 if the device is successfully initialized and provides device information.
- Returns an error code otherwise, and no device information is returned.

4.5. Init Function

This method is used to initialize devices. If USB permission is not granted, the function first requests USB permission before proceeding to initialize the device. Only one device can be initialized at a time.

JAVA:

public int Init(final DeviceModel name, DeviceInfo deviceInfo)

PARAMETERS:

- **name:** [IN] Device model name.
- **deviceInfo:** [OUT] DeviceInfo object for retrieving device information.

RETURN:

- Returns 0 if the device is successfully initialized and provides device information.
- Returns an error code otherwise, and no device information is returned.

4.6. GetSDKVersion Function

This method is used to retrieve the current running SDK version.

JAVA:

public String GetSDKVersion();

RETURN:

Returns the SDK version as a string.

4.7. IsDeviceConnected Function

This method is used to check whether a device is connected or not.

JAVA:

Public boolean IsDeviceConnected(DeviceModel name);

PARAMETERS:

- **name** = [IN] Device model name.

RETURN:

- Returns **true** if the device is connected.
- Returns **false** otherwise.

4.8. StartCapture Function

This method starts the fingerprint capture process on a device. It should be used only after the fingerprint scanner has been set up. Once initiated, a preview is shown via onPreview(), and the capture process begins. If the capture is successful, onComplete() is triggered with a success message.

JAVA:

public int StartCapture(int minQuality, int timeOut);

PARAMETERS:

- **minQuality:** [IN] Minimum quality to detect the captured fingerprint. Range is 1 – 100.
- **timeOut:** [IN] Timeout value in milliseconds. Setting 0 means infinite timeout.

RETURN:

- Returns 0 if the capture starts successfully.
- Returns an error code otherwise.

4.9. AutoCapture Function

This method should be used after the device has been successfully initialized. It synchronously captures a fingerprint and provides quality metrics upon completion. The capture process begins when the onPreview() callback is triggered, and the function returns both the fingerprint quality score and the NFIQ score.

JAVA:

public int AutoCapture(int minQuality, int timeOut, int[] Quality, int[] NFIQ);

PARAMETERS:

- **minQuality:** [IN] Minimum quality required to capture the fingerprint. Range is 1 – 100.
- **timeOut:** [IN] Timeout value in milliseconds. If the timeout is less than 10000 milliseconds, it is automatically set to 10000 milliseconds.
- **Quality:** [OUT] Array to be filled with the fingerprint quality score.
- **NFIQ:** [OUT] Array to be filled with the fingerprint NFIQ score.

RETURN:

- Returns 0 if the capture is successful and fills the Quality and NFIQ arrays with the respective scores.
- Returns an error code if the capture fails.

4.10. GetImage Function

This method is used to retrieve the last captured image data.

JAVA:

public int GetImage(byte[] image, int[] imageLen, int compressionRatio, ImageFormat format);

PARAMETERS:

- **image:** [OUT] Array to be filled with the image data according to the specified ImageFormat.
- **imageLen:** [OUT] Array to be filled with the actual length of the image data.
- **compressionRatio:** [IN] Compression ratio for WSQ (1 - 10) and JPEG2000 (1 - 15) image formats. For other image formats, set this value to 0.
- **format:** [IN] Specifies the fingerprint image format.

RETURN:

- Returns 0 if the image retrieval is successful, and fills the image and imageLen arrays with the image data and its length.
- Returns an error code if the image retrieval fails.

4.11. StopCapture Function

This method is used to forcefully stop a fingerprint capture that was initiated using StartCapture or AutoCapture.

JAVA:

public int stopCapture();

RETURN:

- Returns 0 if the capture is successfully stopped.
- Returns an error code otherwise.

4.12. GetErrorMessage Function

This method retrieves an error message corresponding to a given error code.

JAVA:

public String GetErrorMessage(int errorCode);

PARAMETERS:

errorCode: [IN] The error code for which the error message is to be retrieved.

RETURN:

Returns the error message associated with the provided error code.

4.13. SetLogProperties Function

This method sets the SDK log level and specifies the file where the logs should be written. The default log level is OFF.

JAVA:

public void SetLogProperties(String file, LogLevel level);

PARAMETERS:

- **file:** [IN] The full path and name of the file where logs will be written. Ensure the application has the required write storage permission.
- **level:** [IN] The log level to write, which determines the depth of the logs. The default log level is OFF.

4.14. Dispose Function

This method releases all allocated memory, data, and unregisters the USB broadcast receiver.

JAVA:

public void Dispose();

5. MFScan_Callback Interface

5.1. OnDeviceDetection

This interface is called when an MFS500, MARC10, MELO31, MARC31, MFS200, or MELO30 device is connected or disconnected.

JAVA:

Void onDeviceDetection (String devicename, DeviceDetection detection);

PARAMETERS:

- **devicename:** [OUT] The name of the connected or disconnected device.
- **detection:** [OUT] The status indicating whether the device is connected or disconnected.

5.2. OnPreview

This interface is called when StartCapture or AutoCapture successfully starts capturing. It provides details about the capture preview.

JAVA:

Void onPreview(int errorCode, int quality, byte[] image);

PARAMETERS:

- **errorCode:** [OUT] 0 indicates success; otherwise, it provides an error code.
- **quality:** [OUT] The quality of the preview fingerprint image.
- **image:** [OUT] The preview image data.

5.3. OnComplete

Only Start capture successfully completes on call this interface.

JAVA:

void onComplete (int errorCode, int Quality, int NFIQ);

PARAMETERS:

- **errorCode:** [OUT] 0 indicates success; otherwise, it provides an error code.
- **quality:** [OUT] The quality of the preview fingerprint image.
- **NFIQ:** [OUT] Fingerprint NFIQ score.

6. Enums

6.1. DeviceModel

```
Public enum DeviceModel
{
   MFS500 ("MFS500"),
   MELO31 ("MELO31"),
   MARC10("MARC10");
}
```

6.2. ImageFormat

```
Public enum ImageFormat
{
     BMP(0),
     RAW(1);
}
```

6.3. LogLevel

```
Public enum LogLevel
{
   OFF(0),
   ERROR(1),
   INFO(2),
   DEBUG(3);
}
```

6.4. DeviceDetection

```
Public enum DeviceDetection
{
   DISCONNECTED(0),
   CONNECTED(1);
}
```

7. Model Class

7.1. DeviceInfo

Property	Description
Make	The manufacturer of the scanner.
Model	The specific model of the scanner.
SerialNo	A unique identifier assigned to the specific scanner during manufacturing.
Firmware	The version of the software embedded in the scanner that controls its functionality.
Width	The physical width of the scanner.
Height	The physical Height of the scanner.
DPI	The physical Dpi of the scanner.

7.2. DeviceList

Property	Description
Model	The specific model of the scanner.