

MIScan SDK Android

Mantra Softech India Pvt. Ltd.

V 1.0.0.0

Contents

CONTROL SHEET	2
DOCUMENT HISTORY	2
1. Overview	3
2. Supported Devices.....	3
3. Dependency	3
4. Functions.....	3
4.1. MIScan Constructor	3
4.2. GetConnectedDevices Function	3
4.3. GetSupportedDevices Function	4
4.4. Init Function	4
4.5. GetSDKVersion Function	4
4.6. IsDeviceConnected Function	5
4.7. StartCapture Function	5
4.8. AutoCapture Function	5
4.9. GetImage Function	6
4.10. StopCapture Function	6
4.11. GetErrorMessage Function	6
4.12. SetLogProperties Function	7
4.13. Dispose Function	7
5. MIScan_Callback Interface.....	8
5.1. onDeviceDetection	8
5.2. onPreview	8
5.3. onComplete	8
6. Enums	9
6.1. DeviceModel	9
6.2. ImageFormat	9
6.3. Device detection	9
6.4. LogLevel	9
7. Model Class	10

7.1. DeviceInfo 10

7.2. DeviceList 10

7.3. IrisAnatomy 10

CONTROL SHEET

Version	1.0.0.0
Release Date	09 Aug, 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

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 iris from the below supported devices.

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

2. Supported Devices

- MIS100V2

3. Dependency

SDK used for the following dependency is required:

- USB Host Enable Android Handset
- Android OS 5.0 and above
- MIScan.aar with 2 architectures “armeabi-v7a” and “arm64-v8a”

4. Functions

4.1. MIScan Constructor

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

JAVA:

```
public MIScan (@NonNull Context context, MIScan\_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 determine the connection status of a specific device.

JAVA:

```
Public int GetConnectedDevices(List<String>connectedDevices);
```

PARAMETERS:

connectedDevices = [OUT] This method retrieves a 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. GetSupportedDevices 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

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.

4.5. 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.

4.6. 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.

4.7. StartCapture Function

This method starts the iris capture process on a device. It should be used only after the iris 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 iris. 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.

4.8. AutoCapture Function

This method should be used after the device has been successfully initialized. It synchronously captures a iris and provides quality metrics upon completion. The capture process begins when the [onPreview\(\)](#) callback is triggered, and the function returns the iris quality score and iris detail like a radius and x, y, value of iris's pupil.

JAVA:

```
public int AutoCapture(int timeOut, int[] Quality, IrisAnatomy irisAnatomy);
```

PARAMETERS:

- **timeOut** = [IN] Timeout value in milliseconds. If the timeout is set to less than 10,000 milliseconds, it will automatically be set to 10,000 milliseconds.
- **Quality** = [OUT] Array to be filled with the iris quality score.
- **irisAnatomy** = [OUT] Object to be filled with the iris anatomy class.

RETURN:

- Returns 0 if the capture starts successfully.
- Returns an error code if the capture fails to start.

4.9. 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 image formats, set this value to 0.
- **format:** [IN] Specifies the iris 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.

4.10. StopCapture Function

This method is used to forcefully stop a iris 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.

4.11. 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.

4.12. **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.

4.13. **Dispose Function**

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

JAVA:

```
public void Dispose();
```


5. MIScan_Callback Interface

5.1. onDeviceDetection

This interface is called when the MIS100V2 device is detected.

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, IrisAnatomy anatomy);
```

PARAMETERS:

- **ErrorCode** = [OUT] 0 indicates success; otherwise, it indicates an error.
- **Quality** = [OUT] The quality of the preview iris image.
- **Image** = [OUT] The preview image data.
- **anatomy** = [OUT] The IrisAnatomy object containing details about the iris anatomy.

5.3. onComplete

Only Start capture successfully completes on call this interface.

JAVA:

```
Void onComplete (int ErrorCode, int Quality, byte[] Image, IrisAnatomy anatomy);
```

PARAMETERS:

- **ErrorCode** = [OUT] 0 indicates success; otherwise, it indicates an error.
- **Quality** = [OUT] The iris quality score.
- **Image** = [OUT] The final image data.
- **anatomy** = [OUT] The IrisAnatomy object containing details about the iris anatomy.

6. Enums

6.1. DeviceModel

```
Public enum DeviceModel
{
    MIS100V2("MIS100V2");
}
```

6.2. ImageFormat

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

6.3. Device detection

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

6.4. LogLevel

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

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.

7.3. IrisAnatomy

Property	Description
IrisX	Iris pupil X Coordinate.
IrisY	Iris pupil Y Coordinate.
IrisR	Iris pupil radius Coordinate.