Content

Before You Begin

Biometrics Overview

Biometrics is a method of recognizing a person based on physical or behavioral characteristics. Biometric information that is used to identify people includes fingerprint, voice, face, iris, handwriting, and hand geometry.

There are two key functions offered by a biometric system. One method is identification, a one-to-many (1:N) matching process in which a biometric sample is compared sequentially to a set of stored samples to determine the closest match. The other is verification, a one-to-one (1:1) matching process in which the biometric system checks previously enrolled data for a specific user to verify whether the user is who he or she claims to be. The verification method provides the best combination of speed and security, especially where multiple users are concerned, and requires a user ID or other identifier for direct matching.

With an increasing reliance on online and mobile technology and other shared resources, more and more transactions of all types are initiated and completed online and remotely. This unprecedented growth in electronic transactions has underlined the need for a faster, more secure and more convenient method of user verification than passwords can provide. Using biometric identifiers offers advantages over traditional methods. This is because only biometric authentication is based on the identification of an intrinsic part of a human being. Tokens such as smartcards, magnetic stripe cards and physical keys, can be lost, stolen, duplicated or left behind. Passwords can be forgotten, shared, hacked or unintentionally observed by a third party. By eliminating these potential trouble spots, biometric technology can provide greater security, with convenience, needed for today's complex electronic landscape.

Advantages of Using Fingerprints

The advantages of using fingerprints include widespread public acceptance, convenience and reliability. It takes little time and effort to scan one's fingerprint with a fingerprint reader, and so fingerprint recognition is considered among the least intrusive of all biometric verification techniques. Ancient officials used thumbprints to seal documents thousands of years ago, and law enforcement agencies have been using fingerprint identification since the late1800s. Fingerprints have been used so extensively and for so

long, there is a great accumulation of scientific data supporting the idea that no two fingerprints are alike.

About Aratek

Aratek has been in the business of helping millions manage their digital identity throughout the globe for more than 14 years. We are dedicated to provide cost-effective products and solutions for governments and organizations with sophisticated end-to-end product portfolio ranging from software to fingerprint scanners to multi-functional biometrics terminals.

With our professional and experienced team, we are proud to offer:

- Complete and cost-effective product line
- Large scale manufacturing capacity
- Fast deployment capability
- Flexible specification configuration

Aratek Copyright Declaration

contents of this manual are for reference only, subject to change without notice. Any reference to third-party companies and products is for demonstration purposes only, and does not constitute acceptance or recommendation. Aratek is not responsible for the performance or use of these third party products. Aratek will make every effort to ensure the accuracy of this document, and will not assume any responsibility or obligation for any errors or inaccuracies that may occur therein.

Technical support

Please login to the official website: http://www.aratek.co to get more technical support.

Feedback

Although we have audited and tested the document before it was published, if you find any errors, omissions, or better suggestions during use, please contact us:

support@aratek.co

Address: 2F, T2-A Building, Shenzhen Software Park, Shenzhen, China.

Telephone: +86-755-26719975

1 System Overview

The Aratek TrustFinger™ SDK is the one-to many (1:N) matching engine software developer's kit that enables programmers to develop extremely fast, highly accurate fingerprint searching programs for use in large scale fingerprint databases.

The TrustFinger™ SDK can be used for two types of applications:

- To identify unknown individuals by matching fingerprints in a fingerprint database (e. g., searching for missing children, criminal investigations, etc.)
- To replace identification codes with a high security, user-friendly method (e.g., time and attendance systems, member management systems, system login without ID)

The TrustFinger™ SDK supports quick and easy 1:N matching system integration in any fingerprint database application where accuracy and search speed are paramount.

Features of TrustFinger™ SDK:

- Succinct and Powerful APIs

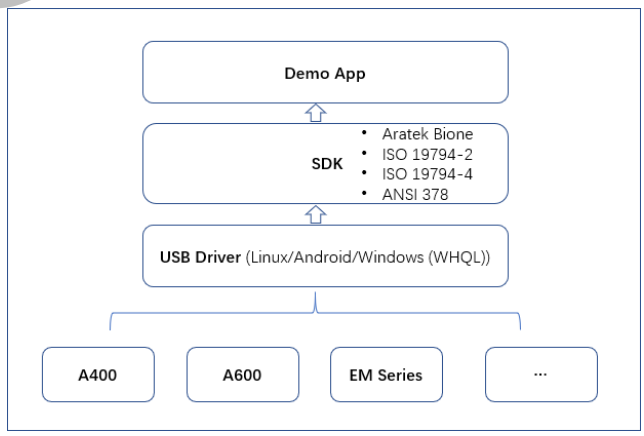
Offers succinct APIs for fingerprint registration and searching so that programmers can easily build fingerprint search systems quickly.

- High accuracy in fingerprint matching

Provides accurate candidate lists with corresponding confidence levels

- High-speed fingerprint searching

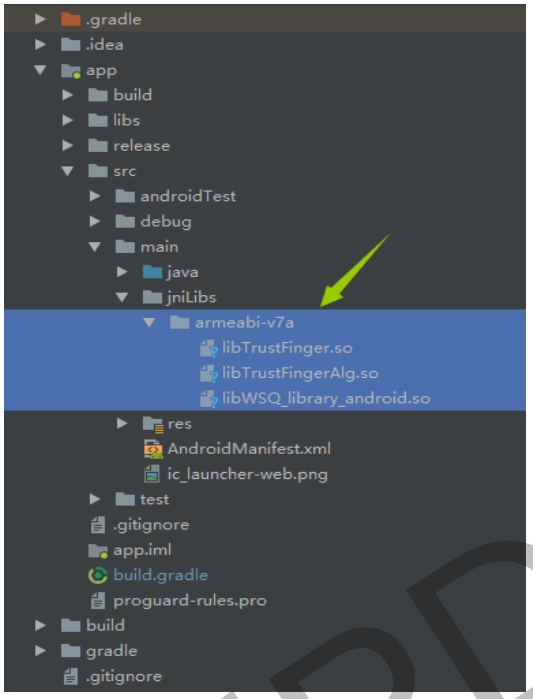
Utilizes an innovative indexing-based algorithm that is different from sequential comparison and that increases the search speed over a mass volume of fingerprints.



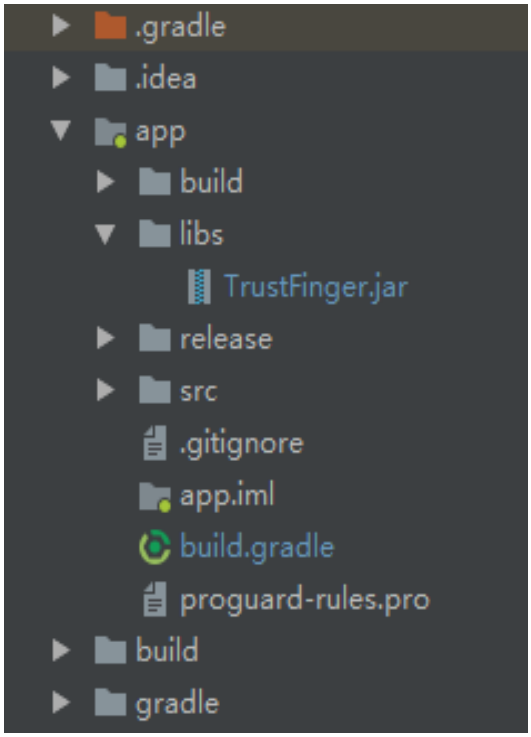
2 Quick Start

2.1 Setup an Android Studio Project with TrustFinger™ Android SDK

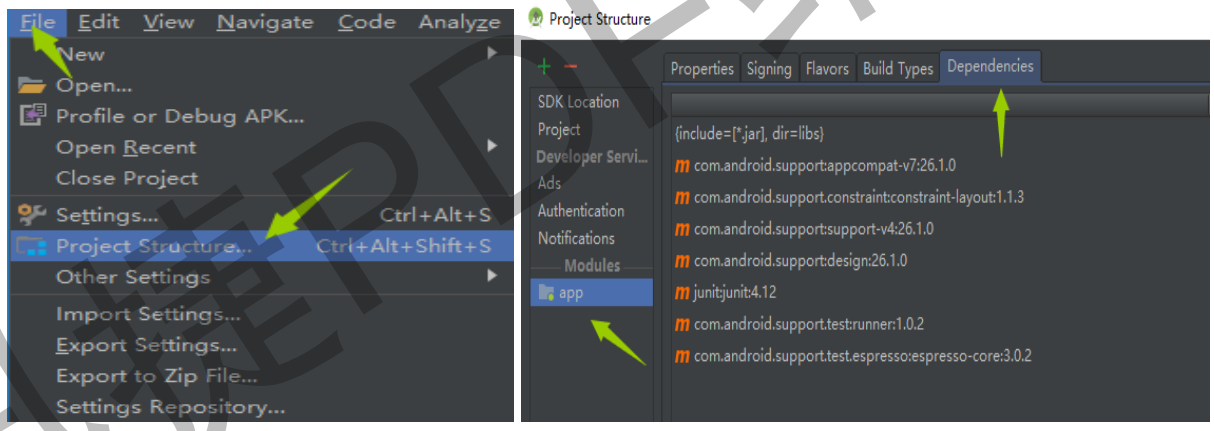
Step 1: Create the folder **/app/src/main/jniLibs**, and then put ***libTrustFinger.so***, ***libTrustFingerAlg.so*** and ***libWSQ_library_android.so*** within their abi folders in that location.



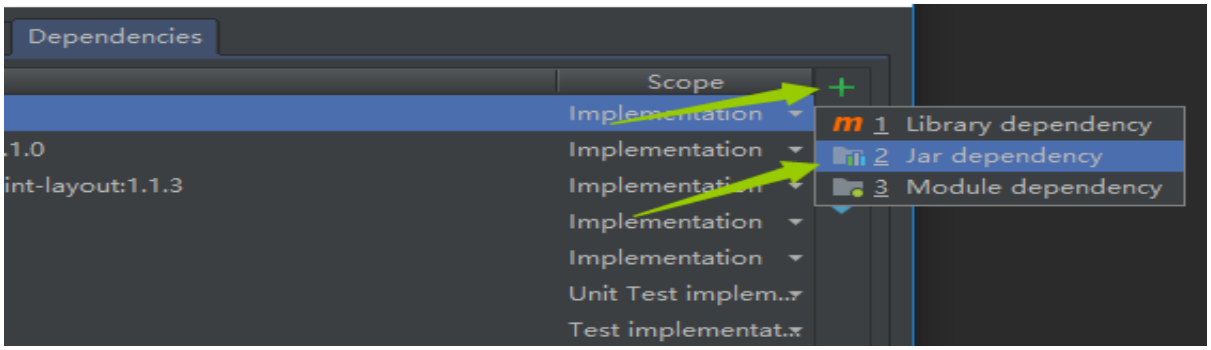
Step 2: Copy the ***TrustFinger.jar*** file into the libs folder under app folder of your project.



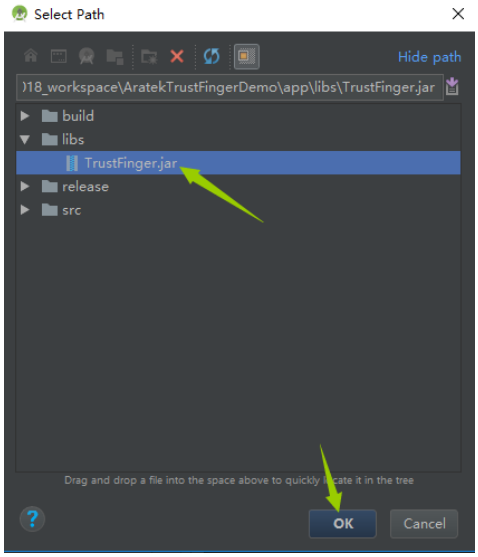
Step 3: Click on **File > Project Structure >Select app > Dependencies** Tab.



Step 4: Click on (+) plus button given on right side and select **Jar Dependency**.



Step 5: This will pop up a dialog box for selecting path. Under this open **libs** folder and add ***TrustFinger.jar***.



Step 6: Once you select the ***TrustFinger.jar*** file then click the **OK** button and your Gradle will Start building.

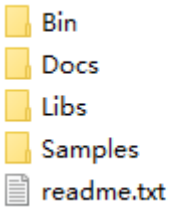
Important Notes:

*If you are not able to find libs then you must be viewing your project in "**Android**" view. On the right side of Android tab you will see the < > symbol. Click on it and select **Project**. This option is available in Android Studio 2.1.*

*If you are using any older version where you can't find this, then you simply need to click on **Android** tab and it will show you a drop down list from that list select **project**.*

2.2 Directory Description

The SDK folder has the following folders:



Bin	- demo apk
Docs	- developer user manual
Libs	- libraries (*.jar and *.so files)
Samples	- demo project source code
Readme.txt	- document structure and version update information.

3 Application Development

A basic application development using the TrustFinger™ SDK can be done by following these steps:

3.1 Fingerprint Image Collection

- 1. Call getInstance() to get device instance, then call initialize() to initialize the SDK. This process needs to be called only once, and is recommended to be called at the start of the application. However, if you have previously called getInstance() and initialize() (i.e. during fingerprint enrollment or fingerprint match as described in Sections 3.2 and 3.3), there is no need to initialize the SDK again.

```
TrustFinger mTrustfinger = TrustFinger.getInstance(getApplicationContext());
mTrustFinger.initialize();
```

- 2. Call openDevice() to open device.

```
int deviceIndex = 0;
TrustFingerDevice mTrustFingerDevice = null;
mTrustFinger.openDevice(deviceIndex, new DeviceOpenListener(){
    @Override
    public void openSuccess(TrustFingerDevice device){
        mTrustFingerDevice = device;
    }
    @Override
    public void openFail(String msg){
    }
});
```

- 3. Collect Image.

```
byte[] rawData = mTrustFingerDevice.captureRawData();
```

3.2 Fingerprint Enrollment

- 1. Call getInstance() to get device instance, then call initialize() to initialize the SDK. This process needs to be called only once, and is recommended to be called at the start of the application. However, if you have previously called getInstance() and initialize() (i.e. during fingerprint image collection or fingerprint match as described in Sections 3.1 and 3.3), there is no need to initialize the SDK again.

```
TrustFinger mTrustfinger = TrustFinger.getInstance(getApplicationContext());
mTrustFinger.initialize ();
```

- 2. Call openDevice()to open device.

```
int deviceIndex = 0;
TrustFingerDevice mTrustFingerDevice = null;
mTrustFinger.openDevice(deviceIndex, new DeviceOpenListener(){
    @Override
    public void openSuccess(TrustFingerDevice device){
        mTrustFingerDevice = device;
    }
    @Override
    public void openFail(String msg){
    }
});
```

- 3. Call captureRawData() to collect image raw data.

```
byte[] rawData = mTrustFingerDevice.captureRawData();
```

- 4. Image quality check and extract fingerprint features.

```
int imageQuality = mTrustFingerDevice.rawImageQuality();
byte[] featureData;
if(imageQuality >= 50){
    featureData = mTrustFingerDevice.extractFeature();
}else{
    // back to capture
}
```

- 5. Repeat steps 3 and 4 to get 3 features, then combine them together to form a template.
- 6. Call generalizeTemplate() to combine the 3 features generated in step 5 into a template.

```
byte[] templateData = mTrustFingerDevice.generalizeTemplate(featureData1,
featureData2, featureData3);
```

- 7. Save the template to complete the fingerprint enrollment.

3.3 Fingerprint Match

- 1. Call getInstance() to get device instance, then call initialize() to initialize the SDK. This process needs to be called only once, and is recommended to be called at the start of the application. However, if you have previously called getInstance() and initialize() (i.e. during fingerprint image collection or fingerprint enrollment as described in Sections 3.1 and 3.2), there is no need to initialize the SDK again.

```
TrustFinger mTrustfinger = TrustFinger.getInstance(getApplicationContext());
mTrustFinger.initialize ();
```

- 2. Call openDevice() to open device.

```
int deviceIndex = 0;
TrustFingerDevice mTrustFingerDevice = null;
mTrustFinger.openDevice(deviceIndex, new DeviceOpenListener(){
    @Override
    public void openSuccess(TrustFingerDevice device){
        mTrustFingerDevice = device;
    }
    @Override
    public void openFail(String msg){
    }
});
```

- 3. Call captureRawData() to collect image raw data.

```
byte[] rawData = mTrustFingerDevice.captureRawData();
```

- 4. Image quality check and extract fingerprint features.

```
int imageQuality = mTrustFingerDevice.rawImageQuality();
byte[] featureData;
if(imageQuality >= 50){
    featureData = mTrustFingerDevice.extractFeature();
}else{
    // back to capture
}
```

■ 5. Call verify() to verify two features.

```
VerifyResult verifyResult = mTrustFingerDevice.verify(SecurityLevel.Level4,
featureData1, featureData2);
int similarity = 0;
if(verifyResult.error == 0){
    if(verifyResult.isMatched){
        // match
        similarity = verifyResult.similarity;
    }else{
        // not match
    }
}else{
    // verify failed, error code: verifyResult.error
}
```

This API compares 2 Bione/ISO/ANSI/ compliance features (supports cross compare) and outputs their similarities. SecurityLevel has 5 ranks ranging from level 1 to 5, with level 5 being the most secure. The recommend setting is level 4.

We have run strict tests inside Aratek using a huge database sample, and the following are the SDK performance results:

Aratek TrustFingrt SDK provides 5 security levels. The relationship with Matching threshold, FAR and security levels is as in below table:

Level	FAR(False accept ratio)	Threshold
1	1%	24
2	0.5%	30
3	0.1%	36
4	**0.01%**	**48**
5	0.001%	60

4 API Description

4.1 Class TrustFinger

4.1.1 getInstance(Context context)

Signature

Method	public static TrustFinger getInstance(Context context)

Description

Get single instance of TrustFinger class.

Parameter

Parameter	Description
context	the context for the receiver and USB accesses

Return

Single instance of TrustFinger.

4.1.2 initialize()

Signature

Method	public void initialize() throws TrustFingerException

Description

Initialize the SDK running environment, normally call this API in application process, and only need to call once before releasing the device.

4.1.3 release()

Signature

Method	public void release()

Description

Release the resources applied by initialize.

4.1.4 getDeviceCount()

Signature

Method	public int getDeviceCount() throws TrustFingerException

Description

Counts the number of connected Aratek fingerprint readers. Only the attached devices to which the caller has been granted permission will be counted.

Return

Number of Aratek fingerprint readers counted.

4.1.5 openDevice(int, DeviceOpenListener)

Signature

Method	public void openDevice(final int deviceIndex, DeviceOpenListener deviceOpenListener) throws TrustFingerException

Description

Initialize device asynchronously, given a particular device index. This function returns immediately, and does not wait whether the APK has access permission or not. If the APK does not have access permission, it will request for one, but it terminates immediately after the request. The parameter DeviceOpenListener will receive openSuccess() invoked with a device object. When an error occurs, openFail() will be invoked with the error message that occurred.

Parameter

Parameter	**Description**
deviceIndex	zero-based index of the device
deviceOpenListener	Call back function once opened the device (See Interface DeviceOpenListener)

4.1.6 getSdkVersion()

Signature

Method	public native String getSdkVersion()

Description

Obtains the SDK version information.

Return

SDK Version.

4.1.7 getDeviceList()

Signature

Method	Public List<String> getDeviceList()

Description

Get a list of connected devices.

Return

A list of device.

4.1.8 setDeviceListener(DeviceListener)

Signature

Method	Public void setDeviceListener(DeviceListener deviceListener)

Description

Set up a listener for plugging and unplugging device. When a device is plugged in, the parameter DeviceListener will receive deviceAttached() invoked with a new list of device, When a device is unplugged, the parameter DeviceListener will receive deviceDetached() invoked with a new list of device.

Parameter

Parameter	**Description**
deviceListener	Call back function (See Interface DeviceListener)

4.2 Class TrustFingerDevice

4.2.1 getImageInfo ()

Signature

Method	public ScannerImageInfo getImageInfo()

Description

Get width, height, and resolution info about the image captured from the present reader.

Return

Image info of the device. (See Class ScannerImageInfo)

4.2.2 getDeviceDescription()

Signature

Method	public DeviceDescription getDeviceDescription()

Description

Retrieve detailed device information about particular scanner.

Return

A description of the device. (See Class DeviceDescription)

4.2.3 captureRawData()

Signature

Method	public byte[] captureRawData() throws TrustFingerException

Description

Collect raw image data from fingerprint reader.

Return

Raw image data.

4.2.4 captureRawData(long)

Signature

Method	public byte[] captureRawData(long timeout) throws TrustFingerException

Description

Collect raw image data from fingerprint reader.

Parameter

Parameter	Description
timeout	Timeout

Return

Raw image data.

4.2.5 captureRawDataLfd(int[])

Signature

Method	public byte[] captureRawDatalfd(int[] lfdstatus) throws TrustFingerException

Description

Collect raw image data from fingerprint reader with LFD.

Parameter

Parameter	Description
lfdstatus	The finger status.(See Enumeration LfdStatus)

Return

Raw image data.

4.2.6 captureBmpData()

Signature

Method	Public byte[] captureBmpData() throws TrustFingerException

Description

Capture a frame with bmp format

Return

Bmp format image

4.2.7 captureBmpDataLfd(int[])

Signature

Method	public byte[] captureBmpDatalfd(int[] lfdstatus) throws TrustFingerException

Description

Collect a bmp image data from fingerprint reader with LFD.

Parameter

Parameter	Description
lfdstatus	The finger status.(See Enumeration LfdStatus)

Return

Bmp format image.

4.2.8 captureISOData(FingerPosition, ImgCompressAlg)

Signature

Method	public byte[] captureISOData(FingerPosition fingerPosition, ImgCompressAlg imgCompressAlg) throws TrustFingerException

Description

Collect one image frame using specified ISO standard format.

Parameter

Parameter	Description

fingerPosition	Finger position (See Enumeration FingerPosition)
ImgCompressAlg	Algorithm about compression: (See Enumeration ImgCompressAlg) UnCompressed = 0 BitPacked = 1 WSQ = 2 JPEG = 3 JPEG2000 = 4 PNG = 5

Return

ISO format image.

4.2.9 captureISODataLfd(FingerPosition, ImgCompressAlg,int[])

Signature

Method	public byte[] captureISODataLfd(FingerPosition fingerPosition, ImgCompressAlg imgCompressAlg,int[] lfdStatus) throws TrustFingerException

Description

Collect one image frame using specified ISO standard format with LFD.

Parameter

Parameter	**Description**
fingerPosition	Finger position (See Enumeration FingerPosition)
ImgCompressAlg	Algorithm about compression: (See Enumeration ImgCompressAlg) UnCompressed = 0 BitPacked = 1 WSQ = 2 JPEG = 3 JPEG2000 = 4 PNG = 5
lfdStatus	The finger status.(See Enumeration LfdStatus)

Return

ISO format image.

4.2.10 captureANSIData(FingerPosition, ImgCompressAlg)

Signature

Method	public byte[] captureANSIData(FingerPosition fingerPosition, ImgCompressAlg imgCompressAlg) throws TrustFingerException

Description

Collect an ANSI format image.

Parameter

Parameter	Description
fingerPosition	Finger position index(See Enumeration FingerPosition)
Imgcompressalg	Compression algorithm: (See Enumeration ImgCompressAlg) UnCompressed = 0 BitPacked = 1 WSQ = 2 JPEG = 3 JPEG2000 = 4 PNG = 5

Return

ANSI format fingerprint data.

4.2.11 captureANSIDataLfd(FingerPosition, ImgCompressAlg,int[])

Signature

Method	public byte[] captureANSIDataLfd(FingerPosition fingerPosition, ImgCompressAlg imgCompressAlg, int[] lfdStatus) throws TrustFingerException

Description

Collect an ANSI format image with LFD.

Parameter

Parameter	Description

fingerPosition	Finger position index(See Enumeration FingerPosition)
Imgcompressalg	Compression algorithm: (See Enumeration ImgCompressAlg) UnCompressed = 0 BitPacked = 1 WSQ = 2 JPEG = 3 JPEG2000 = 4 PNG = 5
lfdStatus	The finger status.(See Enumeration LfdStatus)

Return

ANSI format fingerprint data.

4.2.12 captureWSQData()

Signature

Method	public byte[] captureWSQData() throws TrustFingerException

Description

Collect a WSQ format image.

Return

WSQ format image.

4.2.13 captureWSQDataLfd(int[])

Signature

Method	public byte[] captureWSQDataLfd(int[] lfdStatus) throws TrustFingerException

Description

Collect a WSQ format image with LFD.

Parameter

Parameter	**Description**

lfdStatus	The finger status.(See Enumeration LfdStatus)

Return

WSQ format fingerprint data.

4.2.14 getLedStatus(LedIndex)

Signature

Method	public LedStatus getLedStatus(LedIndex ledIndex) throws TrustFingerException

Description

Get LED status. (This function currently only supports A600 reader)

Parameter

Parameter	Description
ledIndex	LED Index(See Enumeration LedIndex)

Return

Status about specified LED. (See Enumeration LedStatus)

4.2.15 setLedStatus(LedIndex, LedStatus)

Signature

Method	public int setLedStatus(LedIndex ledIndex, LedStatus ledStatus) throws TrustFingerException

Description

Set LED status.

Parameter

Parameter	Description
ledIndex	LED index(See Enumeration LedIndex)
ledStatus	LED status (See Enumeration LedStatus)

Return

Return code	Description
0	Succeed to set the LED status
Others	Failed to set the LED status

4.2.16 extractFeature(byte[], FingerPosition)

Signature

Method	public byte[] extractFeature(byte[] rawData, FingerPosition fingerPosition)

Description

Extract feature data from raw image.

Parameter

Parameter	Description
rawData	Image raw data
fingerPosition	Finger position index(See Enumeration FingerPosition)

Return

ARATEK Bione® Format fingerprint feature data.

4.2.17 extractANSIFeature(byte[], FingerPosition)

Signature

Method	public byte[] extractANSIFeature(byte[] rawData, FingerPosition fingerPosition)

Description

Extract ANSI standard fingerprint feature from raw image.

Parameter

Parameter	Description
rawData	Image raw data

fingerPosition	Finger position index(See Enumeration FingerPosition)

Return

ANSI format fingerprint feature data.

4.2.18 extractISOFeature(byte[], FingerPosition)

Signature

Method	public byte[] extractISOFeature(byte[] rawData, FingerPosition fingerPosition) throws TrustFingerException

Description

Extract ISO standard fingerprint feature from raw image.

Parameter

Parameter	Description
rawData	Image raw data
fingerPosition	Finger position index(See Enumeration FingerPosition)

Return

ISO format feature data.

4.2.19 generalizeTemplate(byte[], byte[], byte[])

Signature

Method	public byte[] generalizeTemplate(byte[] featureData1, byte[] featureData2, byte[] featureData3) throws TrustFingerException

Description

Combine 3 features into one template.

Parameter

Parameter	Description
featureData1	Fingerprint feature data
featureData2	Fingerprint feature data

featureData3	Fingerprint feature data

Return

Fingerprint template data.

4.2.20 verify(SecurityLevel, byte[], byte[])

Signature

Method	public VerifyResult verify(SecurityLevel securityLevel, byte[] feartureData1, byte[] feartureData2)

Description

Verify one feature from another (from database). This API has 4 parameters, securityLevel: 1 to 5, with 5 being the most secure; the default setting is 4. The other 2 parameters are template entrance, the format shall be ARATEK Bione®, ANSI or ISO.

If the value of VerifyResult.error is 0, it means no error has occurred, so you can get the similarity value and check if they are matched. Otherwise it means an error occurred and you cannot get similarity or match results.

Aratek TrustFingrt SDK provides 5 security levels. The relationship with Matching threshold, FAR and security levels is as in below table:

Level	FAR(False accept ratio)	Threshold
1	1%	24
2	0.5%	30
3	0.1%	36
4	**0.01%**	**48**
5	0.001%	60

Parameter

Parameter	**Description**
securityLevel	Security Level, has five levels: 1 to 5. Higher, safer. Recommend to use level 4. (See Enumeration SecurityLevel)
feartureData1	Fingerprint feature data

feartureData2	Fingerprint feature data

Return

Verify result. (See Class VerifyResult)

4.2.21 rawToBmp(byte[], int, int, int)

Signature

Method	public byte[] rawToBmp(byte[] rawData, int imageWidth, int imageHeight, int imageResoltion)

Description

Convert RAW image to BMP.

Parameter

Parameter	**Description**
rawData	Fingerprint raw image data
imageWidth	Image width
imageHeight	Image height
imageResoltion	Image resolution

Return

BMP format image.

4.2.22 bmpToRaw(byte[])

Signature

Method	public byte[] bmpToRaw(byte[] bmpData)

Description

Convert BMP image to raw image.

Parameter

Parameter	**Description**

bmpData	Bitmap data, with BMP header

Return

RAW Fingerprint image data.

4.2.23 rawToWsq(byte[], int, int, int)

Signature

Method	public byte[] rawToWsq(byte[] rawData, int imageWidth, int imageHeight, int imageResolution)

Description

Convert RAW fingerprint image data to WSQ format.

Parameter

Parameter	**Description**
rawData	Raw image data
imageWidth	Image width
imageHeight	Image height
imageResoltion	Resolution

Return

WSQ format image data.

4.2.24 rawToANSI(byte[], int, int, int, int, int)

Signature

Method	public byte[] rawToWsq(byte[] rawData, int imageWidth, int imageHeight, int imageResolution, FingerprintPosition fingerprintPosition, ImgCompressAlg imgCompressAlg)

Description

Convert RAW image to ANSI format.

Parameter

Parameter	Description
rawData	Raw image data
imageWidth	Image width
imageHeight	Image height
imageResoltion	Image resolution
fingerprintPosition	Fingerprint position index (See Enumeration FingerPosition)
imgCompressAlg	Compression algorithm: (See Enumeration ImgCompressAlg) UnCompressed = 0 BitPacked = 1 WSQ = 2 JPEG = 3 JPEG2000 = 4 PNG = 5

Return

ANSI format fingerprint image data.

4.2.25 rawToISO(byte[], int, int, int, int, int)

Signature

Method	public byte[] rawToISO(byte[] rawData, int imageWidth, int imageHeight, int imageResolution, int fingerprintPosition, int imgCompressAlg)

Description

Convert RAW image to ISO format image.

Parameter

Parameter	Description
rawData	Raw image data
imageWidth	Image width

imageHeight	Image height
imageResoltion	Image resolution
fingerprintPosition	Finger position index
imgCompressAlg	Compression algorithm

Retrun

ISO format image data

4.2.26 rawDataQuality (byte[])

Signature

Method	public int rawDataQuality(byte[] rawData) throws TrustFingerException

Description

Get raw image quality. The output score is in a range from 0 to 100; higher score means better image quality.

Parameter

Parameter	**Description**
rawData	Raw image data

Return

Fingerprint image quality.

4.2.27 bmpDataQuality(byte[])

Signature

Method	public int bmpDataQuality(byte[] bmpData) throws TrustFingerException

Description

Get BMP image quality. The output score is in a range from 0 to 100; higher score means better image quality.

Parameter

Parameter	Description
bmpData	BMP image data

Return

Fingerprint image quality.

4.2.28 close()

Signature

Method	public void close() throws TrustFingerException

Description

Close fingerprint reader.

4.2.29 setLFDLevel(int)

Signature

Method	public void setLFDLevel(int level)

Description

Set LFD level.(Only supported by A600)

Parameter

Parameter	Description
level	LFD level(See Enumeration LfdLevel)

4.2.30 getLFDLevel ()

Signature

Method	public int getLFDLevel()

Description

Get LFD level. (Only supported by A600)

Return

LFD level. (See Enumeration LfdLevel)

4.3 Class VerifyResult

If the value of VerifyResult.error is 0, it means no error has occurred, so you can get the similarity value and check if they are matched. Otherwise it means an error occurred and you cannot get similarity or match results.

Attribute	Description
int ***error***	Error code. 0 means successful, others means error code
int ***similarity***	Similarity
boolean ***isMatched***	Wether is matched

4.4 Class ScannerImageInfo

Attribute	Description
int ***width***	Width
int ***height***	Height
int ***resolution***	Resolution

4.5 Class DeviceDescription

Attribute	Description
String ***infoVersion***	Device version info
String ***manufacturer***	Manufacture
String ***productName***	Product name
String ***productModel***	Product module
String ***hwVersion***	Hardware version
String ***bootVersion***	Bootloader version
String ***fwVersion***	Firmware version
String ***serialNumber***	Device serial number
int ***imageWidth***	Image width

int ***imageHeight***	Image height
String ***productionDate***	Produced date and time
int ***deviceId***	Device ID
int ***resolution***	Fingerprint reader resolution
boolean **isUSBSupported**	Whether supports USB
boolean ***isUARTSupported***	Whether supports UART
boolean ***isSPISupported***	Whether supports SPI

4.6 Interface DeviceOpenListener

4.6.1 openSuccess(TrustFingerDevice)

Signature

Method	void openSuccess(TrustFingerDevice trustFingerDevice)

Description

Method when a request to open a device is successful

Parameter

Parameter	Description
trustFingerDevice	Device object(See Class TrustFingerDevice)

4.6.2 openFail(String)

Signature

Method	void openFail(String errorMessage)

Description

Method when a request to open a device has failed

Parameters

Parameter	Description
errorMessage	Error message

4.7 Interface DeviceListener

4.7.1 deviceAttached(List<String>)

Signature

Method	void deviceAttached(List<String> mDevices)

Description

Method when a device is plugged in.

Parameter

Parameter	Description
mDevices	A list of devices.

4.7.2 deviceDetached(List<String>)

Signature

Method	void deviceDetached(List<String> mDevices)

Description

Method when a device is unplugged.

Parameters

Parameter	Description
mDevices	A list of devices.

4.8 Enumeration ImgComCompressAlg

Enumeration	Description
UNCOMPRESSED_NO_BIT_PACKING	Uncompressed – no bit packing
UNCOMPRESSED_BIT_PACKED	Uncompressed – bit packed
COMPRESSED_WSQ	Compressed – WSQ
COMPRESSED_JPEG	Compressed – JPEG

COMPRESSED_JPEG2000	Compressed – JPEG2000
PNG	png

4.9 Enumeration LedIndex

Enumeration	Description
RED	Red LED
GREEN	Green LED

4.10 Enumeration LedStatus

Enumeration	Description
OPEN	Open
CLOSE	Close

4.11 Enumeration SecurityLevel

Enumeration	Description
Level5	Level 5
Level4	Level 4
Level3	Level 3
Level2	Level 2
Level1	Level 1

4.12 Enumeration LfdLevel

Enumeration	Description
OFF	Value is 0.
EXTRA_LOW	Value is 1.
LOW	Value is 2.
MEDIUM	Value is 3.
HIGH	Value is 4.
ULTRA_HIGH	Value is 5.

4.13 Enumeration LfdStatus

Enumeration	Description
UNKOWN	Unkown Finger
NORMAL	Normal Finger
FAKE	Fake Finger

4.14 Enumeration FingerPosition

Enumeration	Description
RightThumb	Right thumb
RightIndexFinger	Right index finger
RightMiddleFinger	Right middle finger
RightRingFinger	Right ring finger
RightLittleFinger	Right little finger
LeftThumb	Left thumb
LeftIndexFinger	Left index finger
LeftMiddleFinger	Left middle finger
LeftRingFinger	Left ring finger
LeftLittleFinger	Left little finger
Unknown	Unknown position

4.15 Enumeration TrustFingerException.Type

Enumeration	Description
SUCCESS	Success
FAIL	Fail
DEVICE_NOT_FOUND	Cannot find device
DEVICE_NOT_AUTHORILIZED	Device is not authenticated
DEVICE_NOT_INITIALIZED	Device is not initialized
DEVICE_NOT_CONNECTED	Device is not connected
DEVICE_NOT_OPENED	Cannot open device
DEVICE_GET_INTERFACE_FAIL	Failed to get device interface
DEVICE_GET_ENDPOINT_FAIL	Failed to get (usb) endpoint
DEVICE_GET_CONNECTION_FAIL	Failed to connect to device

DEVICE_NO_USB_HOST_FEATURE	Not supported USB HOST mode
DEVICE_NOT_ACCESSIBLE	Cannot access the device
DEVICE_ALREADY_OPENED	Device has already been opened
DEVICE_ALREADY_CLOSED	Device has already been closed
API_NOT_SUPPORTED	Not supported API
FAKE_FINGER	Fake finger
CAPTURE_FAIL	Failed to capture image
CAPTURE_ERROR	Error occurred during capture process
TRANSFER_PACKET_ERROR	Transaction packet error
TRANSFER_READ_ERROR	Read transaction error
TRANSFER_WRITE_ERROR	Write transaction error
TRANSFER_CONTROL_ERROR	Control transaction error
INVALID_PARAM_VALUE	Invalid parameter
NOT_ENOUGH_MEMORY	Memory is not enough
INVALID_DEVICE_INDEX	Invalid device index
FP_FEATURE_CONVERT_ERROR	Error occurred during feature conversion
FP_BAD_IMAGE	Image quality is poor
FP_INVALID_DATA	Invalid data
INIT_ALGORITHM_ERROR	Failed to initialize algorithm
UNKNOW_TYPE	Unknown exception
GENERALIZE_TEMPLATE_FAIL_NOT_SAME_FINGER	The three features used to generalize aren't from the same finger
UNKNOW_ERROR	Unknown error