ARATEK BIOMETRICS

BMApi FingerPrint

Application Developer Guide

1 Before You Begin

1.1 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.

1.2 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.

1.3 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

1.4 Aratek Copyright Declaration

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

Contents

1 Before You Begin 2
1.1 Biometrics Overview 2
1.2 Advantages of Using Fingerprints 2
1.3 About Aratek 3
1.4 Aratek Copyright Declaration 3
1 System Overview 6
2 Quick Start 7
2.1 Setup an Android Studio Project with BMApi Android SDK 7
2.2 Directory Description 7
3 Application Development 8
3.2.2. FingerprintImage 11
3.2.3. Bione 12
3.2.4. Result 17
4. Code Samples 20
4.1. Device power on and open 20
4.2. Device close and power off 21
4.3. Fingerprint device image geting 21
4.4. Bione fingerprint image feature extraction 22
4.5. Bione fingerprint Feature template generating 22
4.6. Bione fingerprint Verification 23

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 BMApi Android SDK

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

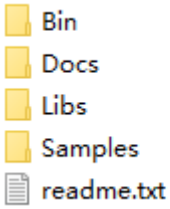
Step 2: Copy the ***.Libs/*.jar*** all jar files into the libs folder under app folder of your project.

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.**

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

3.1. Main Class Table

Package	Introduction	Class	Description
cn.com.aratek.fp	Fingerprint operations	Bione	Fingerprint algorithm
		FingerprintImage	Fingerprint image
		FingerprintScanner	Fingerprint device
cn.com.aratek.util	Smart terminal tool	Result	Error code return

3.2. Main methods

3.2.1. FingerprintScanner

FingerprintScanner includes the below classes：

Method	Description
getInstance	Get FingerprintScanner Class control instantiation
powerOn	Turn off fingerprint sensor
powerOff	Turn on fingerprint sensor
open	Initialize fingerprint sensor
close	Anti-initialize fingerprint sensor
prepare	Prepare to capture fingerprint
finish	Finish capturing fingerprint
getDriverVersion	Get fingerprint sensor driver version
getSN	Get fingerprint sensor SN
getSensorName	Get fingerprint sensor name
capture	Get one frame of fingerprint image
setLfdLevel	Set LFD level

3.2.1.1.getInstance

Function description：Get FingerprintScanner Class control instantiation
Method prototype： public static FingerprintScanner getInstance(Context context);
Parameters： Application context
Return Code： FingerprintScanner control instantiation
Introduction： This class does not provide a constructor and requires a FingerprintScanner class control instance to invoke this method.

3.2.1.2. powerOn

Function description: Power on fingerprint device.
Method prototype: public int powerOn();
Parameters: N/A
Return Code: Error Code (Please refer to FingerprintScanner error code)
Introduction: The third party Android terminal does not need to use this method when using The Aratek fingerprint device, and need to realize the power control by yourself.

3.2.1.3. powerOff

Function description: Power off fingerprint device
Method prototype: public int powerOff()
Parameters: N/A
Return Code: Error Code (Please refer to FingerprintScanner error code)
Introduction: The third party Android terminal does not need to use this method when using The Aratek fingerprint device, and need to realize the power control by yourself.

3.2.1.4. open

Function description: Open and initialize fingerprint device
Method prototype: public int open()
Parameters: N/A
Return Code: Error Code (Please refer to FingerprintScanner error code)
Introduction: N/A

3.2.1.5. close

Function description: Close fingerprint device
Method prototype: public int close()
Parameters: N/A
Return Code: Error Code (Please refer to FingerprintScanner error code)
Introduction: N/A

3.2.1.6. prepare

Function description: Prepare to capture fingerprint
Method prototype: public int prepare()
Parameters: N/A
Return Code: Error Code (Please refer to FingerprintScanner error code)
Introduction: N/A

3.2.1.7.finish

Function description：Fingerprint capturing finished.
Method prototype： public int finish()
Parameters： N/A
Return Code： Error Code (Please refer to FingerprintScanner error code)
Introduction： N/A

3.2.1.8.getDriverVersion

Function description：Get the version of the fingerprint device
Method prototype： public Result getDriverVersion()
Parameters： N/A
Return Code： Including the String type of driver version of the fingerprint device and the Result instance of the error code. (Please refer to the FingerprinScanner Error Code)
Introduction： N/A

3.2.1.9.getSN

Function description：Get serial number of the fingerprint device
Method prototype： public Result getSN()
Parameters： N/A
Return Code： Including the String type of serial number of the fingerprint device and the Result instance of the error code. (Please refer to the FingerprinScanner Error Code)
Introduction： N/A

3.2.1.10.getSensorName

Function description：Get the name of the fingerprint sensor chip
Method prototype： public Result getSensorName()
Parameters： N/A
Return Code： Including the String type of the fingerprint device sensor chip and the Result instance of the error code. (Please refer to the FingerprinScanner Error Code)
Introduction： N/A

3.2.1.11.capture

Function description：capture one frame fingerprint image of FingerprintImage type.
Method prototype： public Result capture()
Parameters： N/A
Return Code： Including the FingerprintImage type of the fingerprint image and the Result instance of the error code. (Please refer to the FingerprinScanner Error Code)
Introduction： N/A

3.2.1.12. setLfdLevel

Function description: Set LFD level.
Method prototype: public int setLfdLevel(int level)
Parameters: LFD level
Return Code: Error code (Please refer to FingerprintScanner Error Code)
Introduction: N/A

3.2.1.13. FingerprintScanner error code

Definition	Error Code	Description
RESULT_OK	0	Operates successfully
RESULT_FAIL	-1000	Operates failed
WRONG_CONNECTION	-1001	Wrong connection of the device
DEVICE_BUSY	-1002	Device is busy
DEVICE_NOT_OPEN	-1003	Device is not open
TIMEOUT	-1004	Timeout
NO_PERMISSION	-1005	Unauthorized
WRONG_PARAMETER	-1006	Wrong parameter
DECODE_ERROR	-1007	Decode error.
INIT_FAIL	-1008	Initialize failed
UNKNOWN_ERROR	-1009	Unknown error
NOT_SUPPORT	-1010	Not supported
NOT_ENOUGH_MEMORY	-1011	Not enough memory
DEVICE_NOT_FOUND	-1012	Cannot find out the device
DEVICE_REOPEN	-1013	Device open repeatedly
INVALID_LICENSE	-1014	Invalid license
USER_ABORT	-1015	Suspended by users
ACCESS_ERROR	-1016	Access error
NO_FINGER	-2005	Cannot detect finger

3.2.2. FingerprintImage

FingerprintImage Class includes the below public members and public methods:

Members or Methods	Description
raw	Fingerprint image raw data（byte[]）
width	Fingerprint image width（int）
height	Fingerprint image height（int）
dpi	Fingerprint image DPI（int）
convert2Bmp	Converts a fingerprint image into a BMP image byte sequence

3.2.2.1. convert2Bmp

Function description: Converts a fingerprint image into a BMP image byte sequence
Method prototype: public byte[] convert2Bmp()
Parameters: N/A
Return Code: byte[] type BMP image byte sequence
Introduction: N/A

3.2.3. Bione

Bione Class includes the below public methods:

Methods	Description
initialize	Initialize Bione algorithm
exit	Exit Bione algorithm and clean up
getVersion	Get the version of the algorithm
getFingerprintQuality	Get the fingerprint image quality of specified FingerprintImage type
extractFeature	Extract features from the fingerprint image of the FingerprintImage type
makeTemplate	Make a template from three different feature of one finger
isFreeID	Determines whether the ID has been registered in the current fingerprint library
getFreeID	Gets an unused ID value from the current fingerprint library
getFeature	Gets the fingerprint feature or template of the specified ID from the current fingerprint library
getAllFeatures	Gets all fingerprint features or templates from the current fingerprint library
getEnrolledCount	Gets the number of registered fingerprint features or templates in the current fingerprint library
enroll	Register fingerprint features into the current fingerprint database
delete	Deletes the fingerprint features of the specified ID in the current fingerprint library
clear	Clear the current fingerprint database
verify	verify the fingerprint features of the ID in the current fingerprint database with the target fingerprint features
verify	Verify two features are matched or not.
identify	Search the id of the matched fingerprint feature from the current fingerprint database and return the result
setSecurityLevel	Sets the security level of the fingerprint verification

3.2.3.1. initialize

Function description: Iinitialize Bione algorithm
Method prototype: public static int initialize(Context context, String dbPath)
Parameters: context is the current application context.
dbPath is the path of fingerprint library file.
Return Code: Error code (Please refer to Bione Error Code)
Introduction: N/A

3.2.3.2.exit

Function description：Exit Bione algorithm and clean up
Method prototype： public static int int exit()
Parameters： N/A
Return Code： Error code (Please refer to Bione Error Code)
Introduction： N/A

3.2.3.3.getVersion

Function description：Get the version of the algorithm
Method prototype： public static int getVersion()
Parameters： N/A
Return Code： Version of the algorithm
Introduction： N/A

3.2.3.4.getFingerprintQuality

Function description：Get the fingerprint image quality of specified FingerprintImage type
Method prototype： public static int getFingerprintQuality(FingerprintImage image)
Parameters： image is the fingerprint image of FingerprintImage type
Return Code： >= 0 Value of fingerprint image quality
<0 Error code (Please refer to Bione Error Code)
Introduction： N/A

3.2.3.5.extractFeature

Function description: Extract features from the fingerprint image of the FingerprintImage type
Method prototype: public static Result extractFeature(FingerprintImage image)
Parameters: image is the fingerprint image of FingerprintImage type
Return Code: including the byte[] fingerprint feature data and the error code (Please refer to Bione Error Code)
Introduction: N/A

3.2.3.6.makeTemplate

Function description：Make a template from three different feature of one finger
Method prototype： public Result makeTemplate(byte[] feature1, byte[] feature2, byte[] feature3)
Parameters： feature1 is the fingerprint feature data of one finger captured at the first time.
Feature2 is the fingerprint feature data of one finger captured at the second time.
Feature3 is the fingerprint feature data of one finger captured at the third time.
Return Code： including the byte[] fingerprint template data and the Result instance of the error code (Please refer to Bione Error Code)
Introduction： N/A

3.2.3.7. isFreeID

Function description: Determines whether the ID has been registered in the current fingerprint library
Method prototype: public static boolean isFreeID(int id)
Parameters: id is the one determined whether can be registered or not.
Return Code: true means that the id hasn't been used and can be registered.
false means that the id has been used and cannot be registered.
Introduction: N/A

3.2.3.8. getFreeID

Function description: Gets an unused ID from the current fingerprint library
Method prototype: public static int getFreeID()
Parameters: N/A
Return Code: >= 0 id can be used to register.
< 0 error code (Please refer to Bione Error Code)
Introduction: N/A

3.2.3.9. getFeature

Function description: Gets the fingerprint feature or template of the specified ID from the current fingerprint library
Method prototype: public static getFeature(int id)
Parameters: id is the one used to get the fingerprint feature
Return Code: including the byte[] of the fingerprint feature or template data and the Result instance of the error code (Please refer to the Bione Error Code)
Introduction: N/A

3.2.3.10. getAllFeatures

Function description: Gets all fingerprint features or templates from the current fingerprint library
Method prototype: public static Result getAllFeatures()
Parameters: N/A
Return Code: including the Map<Integer, byte[]> of the fingerprint feature or template data and the Result instance of the error code (Please refer to the Bione Error Code)
Introduction: The returned Map object includes an Integer type ID value and the corresponding byte[] type feature value

3.2.3.11. getEnrolledCount

Function description: Gets the number of registered fingerprint features or templates in the current fingerprint library
Method prototype: public static int getEnrolledCount()
Parameters: N/A
Return Code: >= 0 the number of the registered fingerprint feature or template in the current fingerprint database
< 0 error code (Please refer to Bione Error Code)
Introduction: N/A

3.2.3.12. enroll

Function description: Register fingerprint features into the current fingerprint database
Method prototype: public static int enroll(int id, byte[] feature)
Parameters: id is the value of ID which needs to be registered to the current fingerprint database.
Feature is the fingerprint feature or template data which need to be registered to the current fingerprint database.
Return Code: error code (Please refer to Bione Error Code)
Introduction: must ensure that the ID value is unique in the current library. If necessary, use the isFreeID method to determine whether the ID has been registered or use the getFreeID method to apply for an unused ID value for registration.

3.2.3.13. delete

Function description: Deletes the fingerprint features of the specified ID in the current fingerprint library
Method prototype: public static int delete(int id)
Parameters: id is the value of ID which needs to be deleted from the current fingerprint database.
Return Code: error code (Please refer to Bione Error Code)
Introduction: N/A

3.2.3.14. clear

Function description: Clear the current fingerprint database
Method prototype: public static int clear()
Parameters: N/A
Return Code: error code (Please refer to Bione Error Code)
Introduction: N/A

3.2.3.15. verify

Function description: verify the fingerprint features of the ID in the current fingerprint database with the target fingerprint features
Method prototype: public static Result verify(int id, byte[] feature)
Parameters: id is the value of ID which needs to be verified in the current fingerprint database.
Feature is the fingerprint feature data which need to be verified.
Return Code: Including the Boolean type verification result and the Result instance of the error code (Please refer to Bione Error Code)
Introduction: N/A

3.2.3.16. verify

Function description: Verify two features are matched or not.
Method prototype: public static Result verify(byte[] feature1, byte[] feature2)
Parameters: feature1 is the first fingerprint feature.
Feature2 is the second fingerprint feature.
Return Code: Including the Boolean type verification result and the Result instance of the error code (Please refer to Bione Error Code)

Introduction： N/A

3.2.3.17. identify

Function description：Search the id of the matched fingerprint feature from the current fingerprint database and return the result

Method prototype： public static int identify(byte[] feature)

Parameters： Feature is the fingerprint feature or template which used to be identified in current database.

Return Code： >= 0 Identifies successfully and returns the Identified ID.

<0 Error code (Please refer to Bione Error Code)

Introduction： N/A

3.2.3.18. setSecurityLevel

Function description：Sets the security level of the fingerprint verification

Method prototype： public static void setSecurityLevel(int level)

Parameters： level is the security level (HIGH, MEDIUM,LOW)

Return Code： N/A

Introduction： N/A

3.2.3.19. Bione Error Code

Definition	Error Code	Description
RESULT_OK	0	Algorithm operates successfully
INITIALIZE_ERROR	-2000	Algorithm operates failed
INVALID_FEATURE_DATA	-2001	Invalid feature format
BAD_IMAGE	-2002	Bad fingerprint image quality
NOT_MATCH	-2003	Not the same fingerprint
LOW_POINT	-2004	low fingerprint match score low when 1:1 or not match when making the template
NO_FINGER	-2005	No finger in the image
NO_RESULT	-2006	No returned result when 1:N
OUT_OF_BOUND	-2007	ID is out of bound (<0 or >=the maximum of the fingerprint database)
DATABASE_FULL	-2008	Fingerprint database is full
LIBRARY_MISSING	-2010	Cannot find out the library.
UNINITIALIZE	-2011	Algorithm is uninitialized.
REINITIALIZE	-2012	Algorithm is initialized repeatly.
REPEATED_ENROLL	-2013	ID has been registered.
NOT_ENROLLED	-2014	ID hasn’t been registered.
FEATURE_CONVERT_ERROR	-2015	Error when converting the feature.

3.2.4. Result

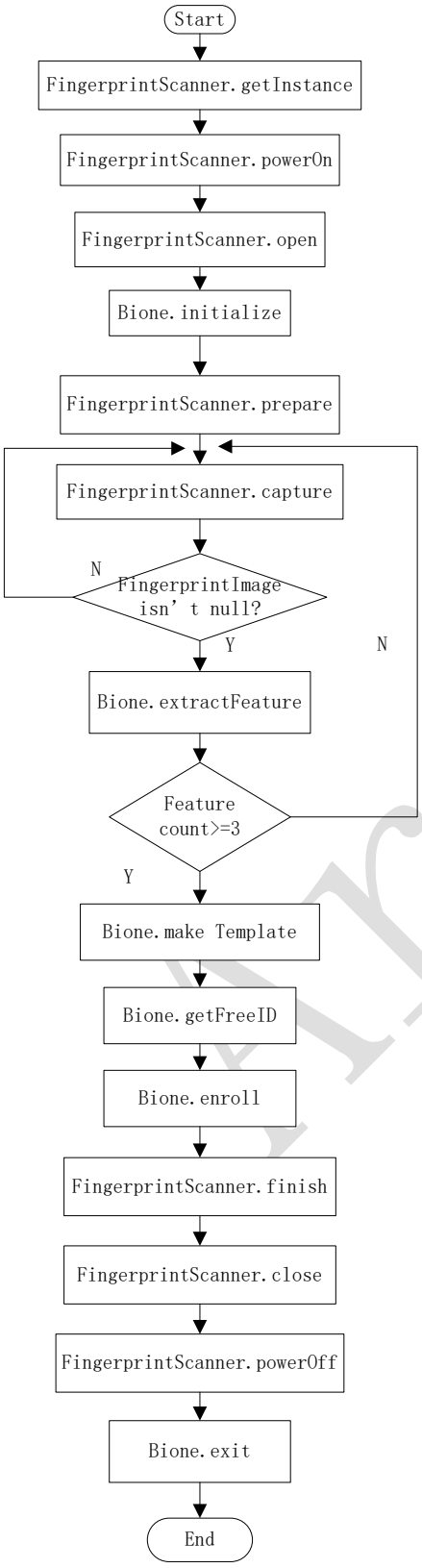
Result Class includes the below members：

Members	Description
error	Error code（int）
data	Returned value（Object）

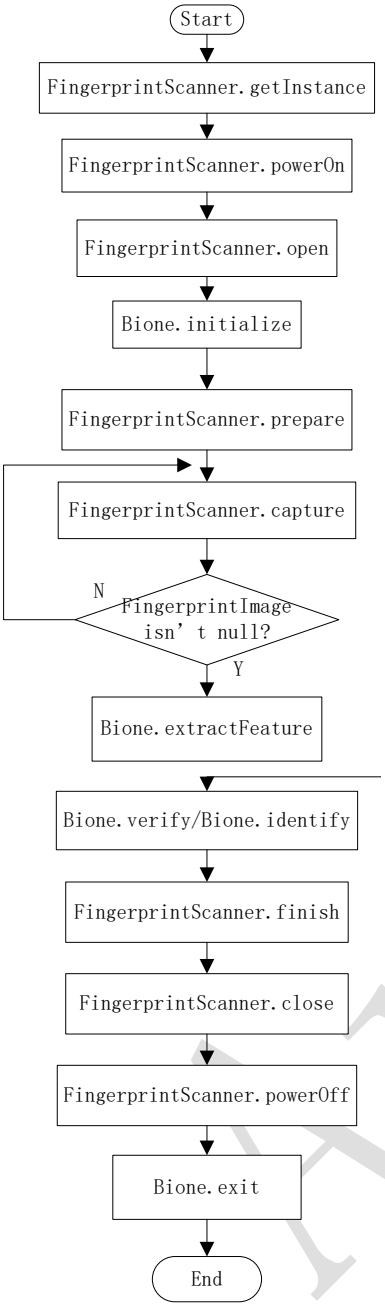
Note: All results containing error codes and returned values of other types will return the Result type uniformly, in which the public member error contains the error codes returned by the operation, and the public member data contains the type data defined by the return value item described in this document. Please cast the data to this type before using.

4. Calling Process of Main Interfaces

4.1. Bione fingerprint enroll process



4.2. Bione fingerprint verify process



4. Code Samples

4.1. Device power on and open

Device needs to be powered on and opened after getting device control instance. These operations will spend some time, so we would suggest you to put the operations on a new thread and hint the user of device powering on and opening. If device is used continuously, there is no need to operate “power on”, “open”, “close” and “power off” every time, you can do the operation only once. Call “close”, “power off” when device is not used for a long time. The following code takes fingerprint API for example, showing us how to get device powered on and initialized

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mScanner = FingerprintScanner.getInstance(this);
    …
}

@Override
protected void onResume() {
    super.onResume();

    int error;
    if ((error = mScanner.powerOn()) != FingerprintScanner.RESULT_OK) {
        Toast.makeText(this, getString(R.string.fingerprint_device_power_on_failed) +
error, Toast.LENGTH_SHORT).show();
    }
    if ((error = mScanner.open()) != FingerprintScanner.RESULT_OK) {
        Toast.makeText(this, getString(R.string.fingerprint_device_open_failed) + error,
Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, getString(R.string.fingerprint_device_open_success),
Toast.LENGTH_SHORT).show();
        enableControl(true);
    }
}
```

4.2. Device close and power off

When device is not used for a long time, call “close” and “power off” to save system resource and power consumption. Following code shows how to call the method

```
@Override
protected void onPause() {
    enableControl(false);
    int error;
    if ((error = mScanner.close()) != FingerprintScanner.RESULT_OK) {
        Toast.makeText(this, getString(R.string.fingerprint_device_close_failed) + error,
Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, getString(R.string.fingerprint_device_close_success),
Toast.LENGTH_SHORT).show();
    }
    if ((error = mScanner.powerOff()) != FingerprintScanner.RESULT_OK) {
        Toast.makeText(this, getString(R.string.fingerprint_device_power_off_failed) +
error, Toast.LENGTH_SHORT).show();
    }
    super.onPause();
}
```

4.3. Fingerprint device image geting

After calling method FingerprintScanner.capture, it will return FingerprintImage type instance, you can call FingerprintImage.convert2Bmp method to convert it into BMP format byte sequence, and call Bione.getFingerprintQuality to get quality score of the fingerprint image, or Bione.extractFeature to make feature data based on the fingerprint image. Sample code as below

```
    mScanner.prepare();
    Result res = mScanner.capture();
    mScanner.finish();
    if (res.error != FingerprintScanner.RESULT_OK) {
        Toast.makeText(this, getString(R.string.capture_image_failed) + res.error,
Toast.LENGTH_SHORT).show();
        return;
    }
    FingerprintImage fi = (FingerprintImage) res.data;
    Log.i(TAG, "Fingerprint image quality is " + Bione.getFingerprintQuality(fi));
```

4.4. Bione fingerprint image feature extraction

Fingerprint feature is a series of representative data points extracted from fingerprint image. Fingerprint image needs to extract feature so that it can be converted to template and be verified. AraBMAPI provides method Bione.extractFeature to extract feature, it needs to input a FingerprintImage type instance (FingerprintScanner.capture method to capture). The extracted feature is in byte type, it can be used as parameter in making template and verifying. Sample code as below

```
    // fi为采集到的FingerprintImage类型的指纹图像
    Result res = Bione.extractFeature(fi);
    if (res.error != Bione.RESULT_OK) {
        Toast.makeText(this, getString(R.string.enroll_failed_because_of_extract_feature) +
res.error, Toast.LENGTH_SHORT).show();
        return;
    }
    byte[] fpFeat = (byte[]) res.data;
```

4.5. Bione fingerprint Feature template generating

Fingerprint template is generated from three finger features extracted from one finger. Input the three extractedfinger features to method Bione.makeTemplate, and the template can be output in byte type array, code below:

```
// fpFeat1、fpFeat2、fpFeat3为采集到的同一手指的三次指纹特征
Result res = Bione.makeTemplate(fpFeat1, fpFeat2, fpFeat3);
if (res.error != Bione.RESULT_OK) {
    Toast.makeText(this, getString(R.string.enroll_failed_because_of_make_template) +
res.error, Toast.LENGTH_SHORT).show();
    return;
}
byte[] fpTemp = (byte[]) res.data;
```

4.6. Bione fingerprint Verification

AraBMAPI provides two ways of matching: 1:1 match Bione.verify method and 1:N identify Bione.identify method. Verification is to perform 1:1 matching: two features match, a feature and a template match, and two templates match. Bione.verify method returns 1:1 verification result (success or failure). Bione.identify method returns the id that best matching in database. Code below:

```
    // fpFeat要进行1:1比对的指纹特征，mId为指纹库里要匹配的指纹ID
    Result res = Bione.verify(mId, fpFeat);
    if (res.error != Bione.RESULT_OK) {
        Toast.makeText(this, getString(R.string.verify_failed_because_of_error) + res.error,
Toast.LENGTH_SHORT).show();
        return;
    }
    if ((Boolean) res.data) {
        Toast.makeText(this, getString(R.string.fingerprint_match),
Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, getString(R.string.fingerprint_not_match),
Toast.LENGTH_SHORT).show();
    }
    // 1:N比对
    int id = Bione.identify(fpFeat);
    if (id < 0) {
        Toast.makeText(this, getString(R.string.identify_failed_because_of_error) + id,
Toast.LENGTH_SHORT).show();
        return;
    }
    Toast.makeText(this, getString(R.string.identify_match) + id,
Toast.LENGTH_SHORT).show();
```