BioMini SDK for Android

V2.0.1

Table of Contents

1. Introduction

The BioMini Android SDK provides the specific functions to capture fingerprints, to display scanner real-time images on the screen while the finger is on the platen.

1.1 What's New

Version 2.0.1

- Application speed improved due to Libusb-1.0 module
- Bugfix.

1.2 History

Version 2.0.0

- SDK renewal – SDK is not compatible with older version due to interface & API change
- S/W LFD supporting device added: BioMini Slim
- Bugfix.

Version 1.4.5

- S/W LFD supporting device added: BioMini Plus2, BioMini Slim2
- New API added
  - UFA_ExtractTemplateFromWSQ()
  - UFA_ExtractTemplateFromBMP()
- Capture speed improved: BioMini Slim2

Version 1.4

- Supporting device added: BioMini Slim2
- BioMini CardSDK interface module added
- Compatible with new Suprema fingerprint scanners: BioMini Slim 2 (BM-Slim2)
- Redefine the followings from UFA_XXX to ENUM type

    - ECODE
    - PARAM
    - TEMPLATE_TYPE
    - FRAME_RATE
    - SCANNER_OPTIONS
- Added new parameter
    - Enable auto sleep mode - UFA_PARAM_ENABLE_AUTOSLEEP
- Added new functions
    - UFA_SetSleep
    - UFA_IsAwake
    - UFA_GetCompanyID
    - UFA_GetCoreCoordinate

Version 1.3

- Supporting device added: BioMini Plus2
- Compatible with new Suprema fingerprint scanners: BioMini Plus2 (SFR550)
- Changed the development tool from ADT/Eclipse to Android Studio/Gradle
- Added new parameter as below
    - Support 360-degree matching - UFA_PARAM_AUTO_ROTATE parameter
- Added new functions as below
    - UFA_SetDevice
    - UFA_GetCaptureImageBufferToBMPBuffer
    - UFA_GetCaptureImageBufferToWSQBufferVar
    - UFA_GetCaptureImageBufferToWSQBufferVarEx
    - UFA_GetFeatureNumber
    - UFA_GetFPQuality

Version 1.2

- Compatible with new Suprema fingerprint scanners: BioMini and SFU-410
- BioMiniAndroid - Added new functions as below
  - Save capture image data of bmp format
  - Set device callback
  - Get version string
  - Set capture frame rate
  - Check scanner capture status
  - IBioMiniDeviceCallback - Abstract class for callback to check device attached

Version 1.1

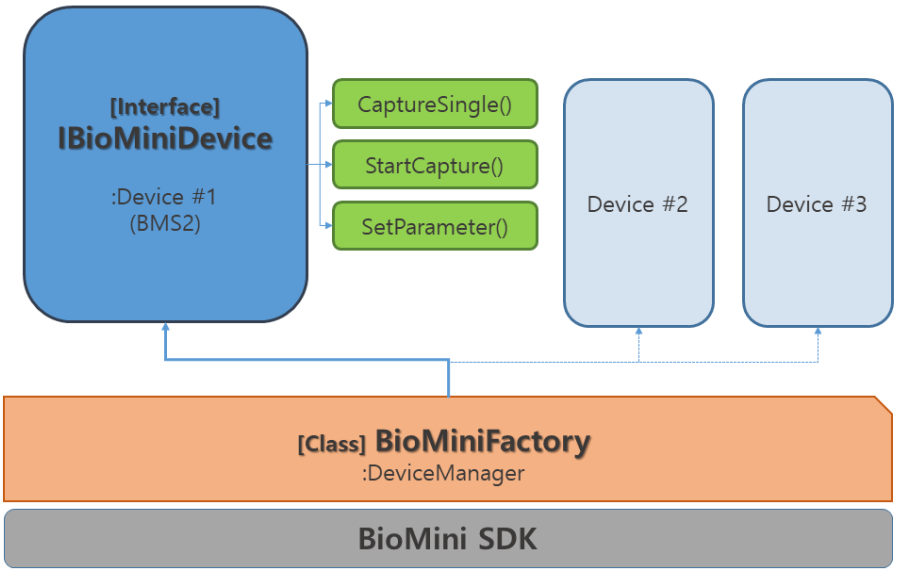
- Add NIST certified WSQ format - UFA_GetCaptureImageBufferToWSQBuffer
- Main class name typo fixed

Version 1.0

- BioMini SDK for Android release
- Supporting device added: BioMini Slim
- Googleplay Demo release
- Support BioMini Slim, BioMini Combo(fingerprint scanner only) and SFU-S20
- BioMiniAndroid - Managing scanner, fingerprint template extraction and 1:1 matching
- Added fingerprint image capturing function
- Added fingerprint template extraction function
- Added 1:1 matching function

1.3 Overview

SDK Structure



BioMini Android SDK consists of the following

- BioMiniFactory class that manages BioMini equipment, Instance and manages actual device handler.
- IBioMiniDevice interface that extracts from instanceized BioMiniFactory and controls actual machine operation
- BioMini SDK for Android is based on Factory pattern.

BioMini Device instance (Handler) matches with each BioMini device and each Instance operates separately.

SDK Package Consists of

Path : /	File Name	Description
	/bin/biominisample.apk	Sample APK package with latest SDK
	/doc/xxxxxx.pdf	Manual Document
	/lib/libBioMini.aar	This version of the Android library
	/sample	Android studio sample directory
	/sample/biominisample/src/main/res/layout.xml	Layout xml file mapped to MainActivity
	/sample/biominisample/src/main/java/com/suprema/biominisample/MainActivity.java	MainActivity of Sample application

System Requirements

The following minimum system requirements are necessary in order to use the SDK described in this document

- **Operating System**:
  - Android 4.1 (Jelly Bean / API Level 17) or above

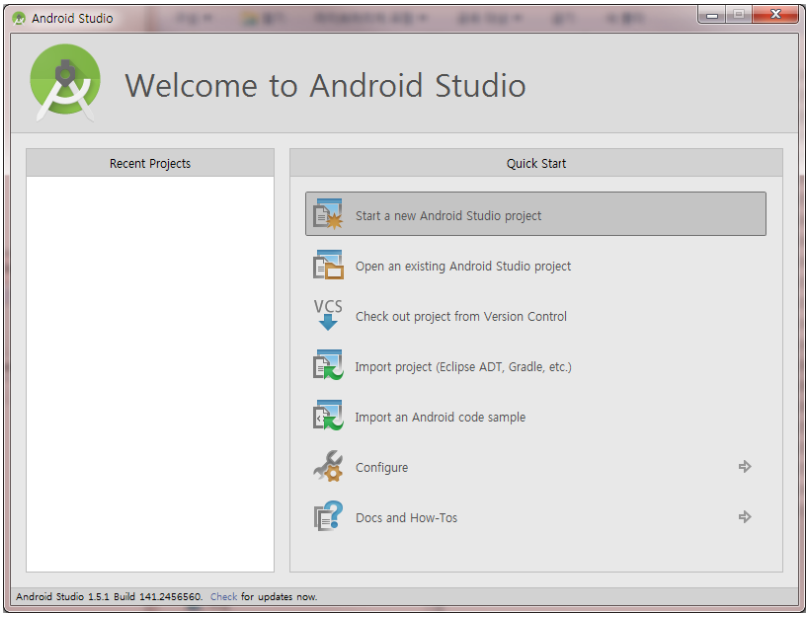
- **Hardware**:
  - Cpu : Quad-Core 1.7GHz or better
  - Memory: 2G or more memory
  - Supported platform : ARMeabi / ARMeabi-V7a / x86
  - Supported programming language : Java (Java library with JNI)
  - Supported scanner : BioMini Slim 2, BioMini Plus 2, BioMini Slim, BioMini
  - **Android USB Host API**

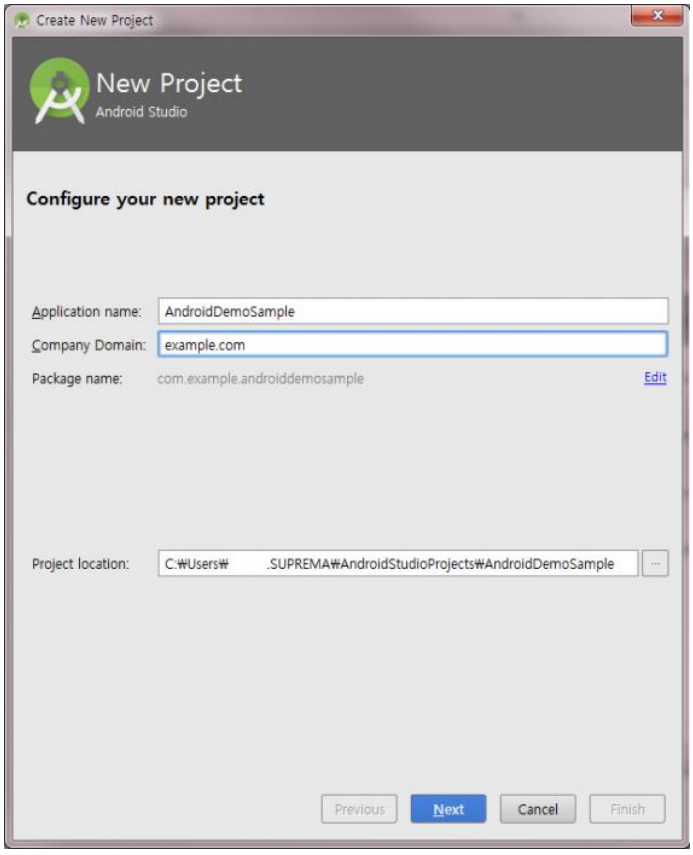
2. Getting Started

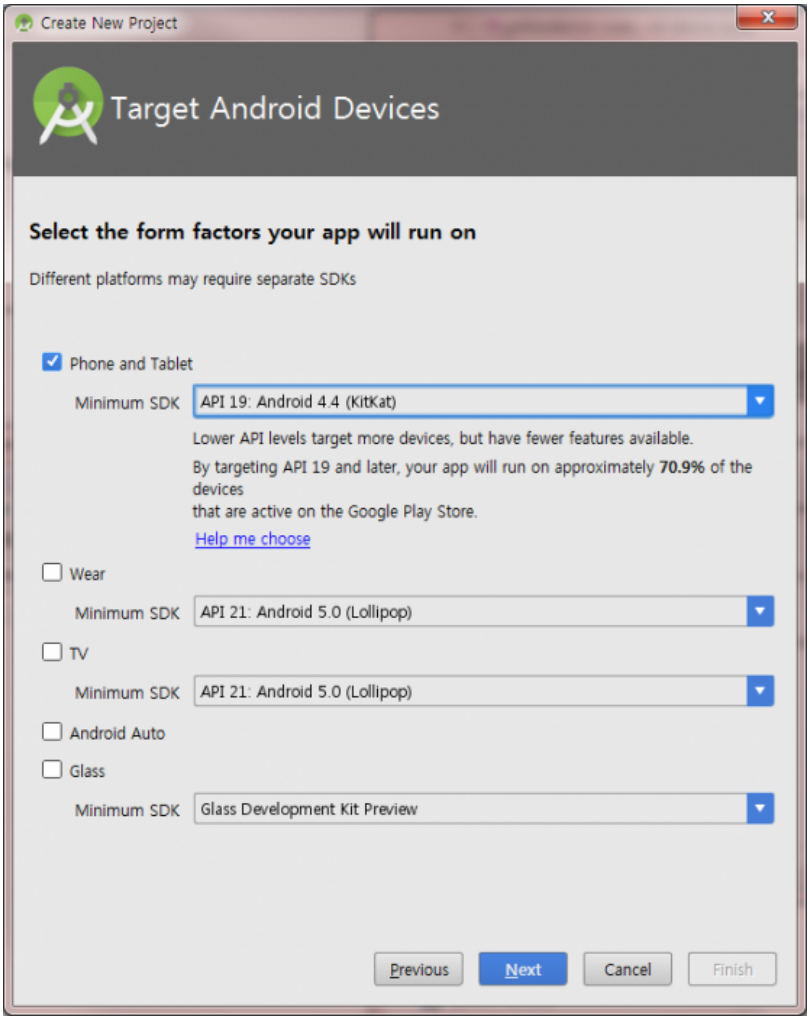
2.1 Environment Setting

Project generation

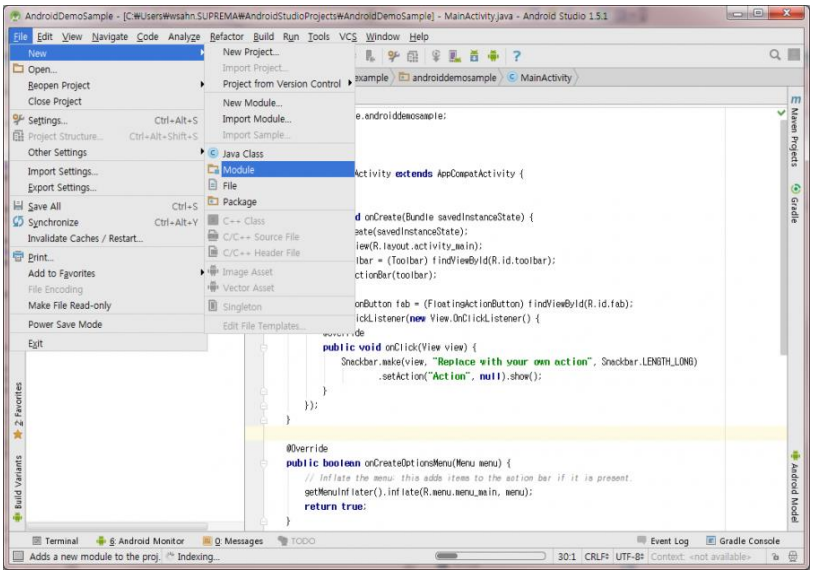
1. Generate a new application project in Android Studio.



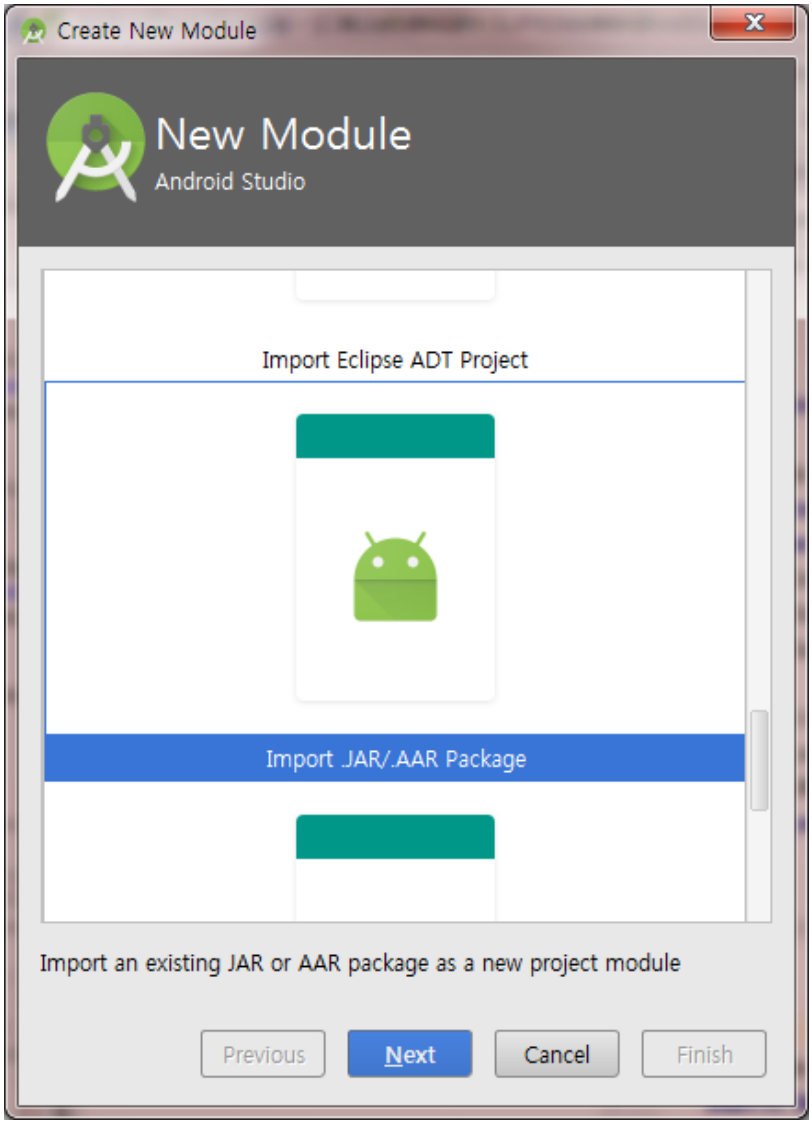




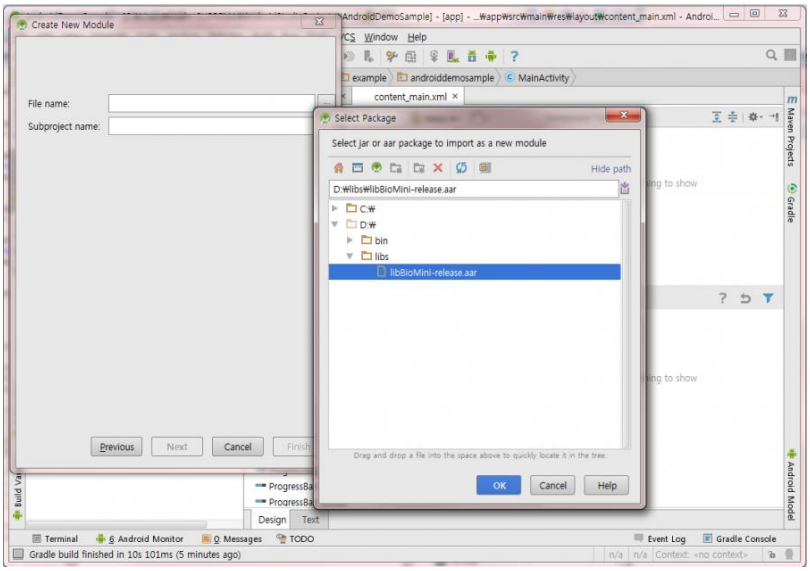
2. Execute File → New → Module menu.



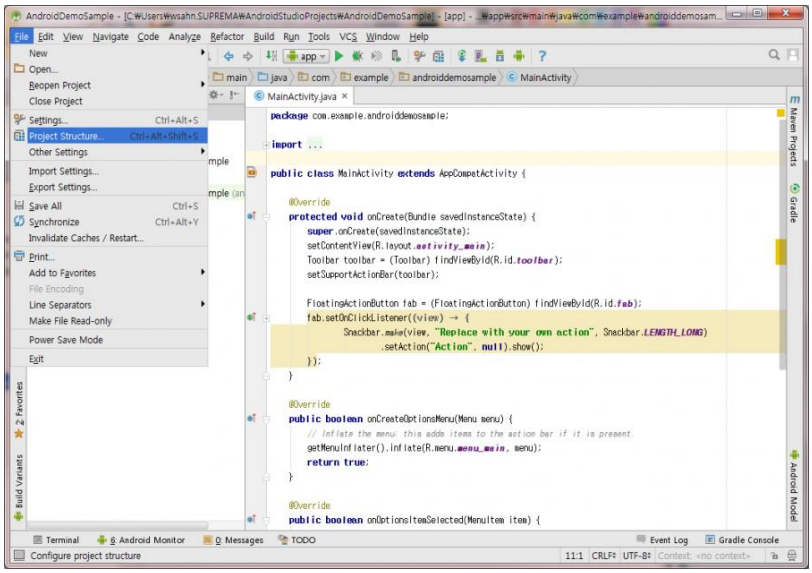
3. Select “Import .JAR/.AAR Package” type and clik Next button.



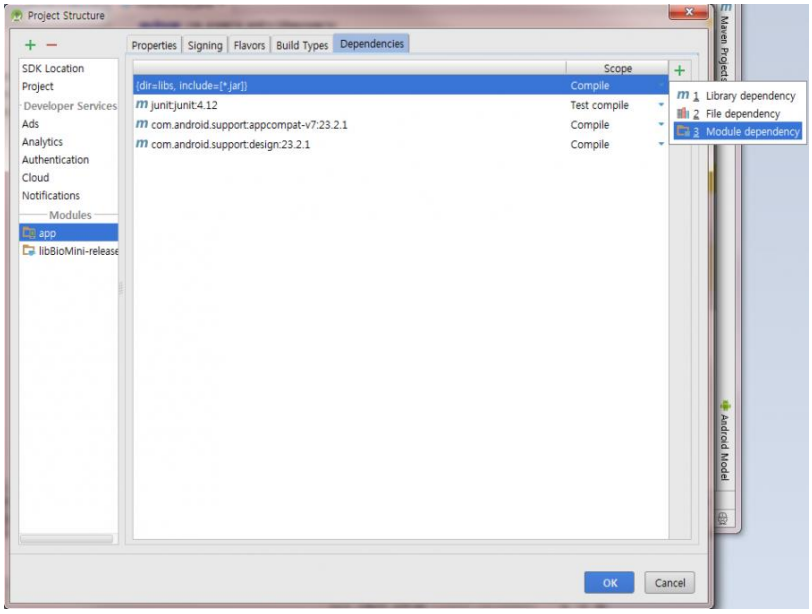
4. Select BioMini Android SDK (libBioMini_v20.aar) in File name and click Finish button to close.



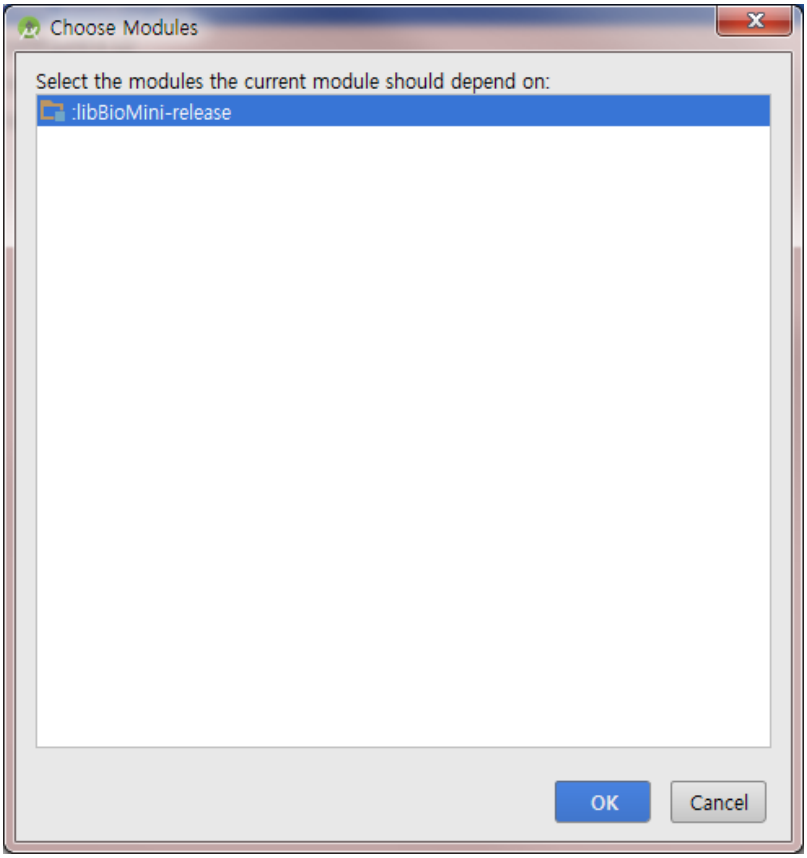
5. Execute File → Project Structure menu.



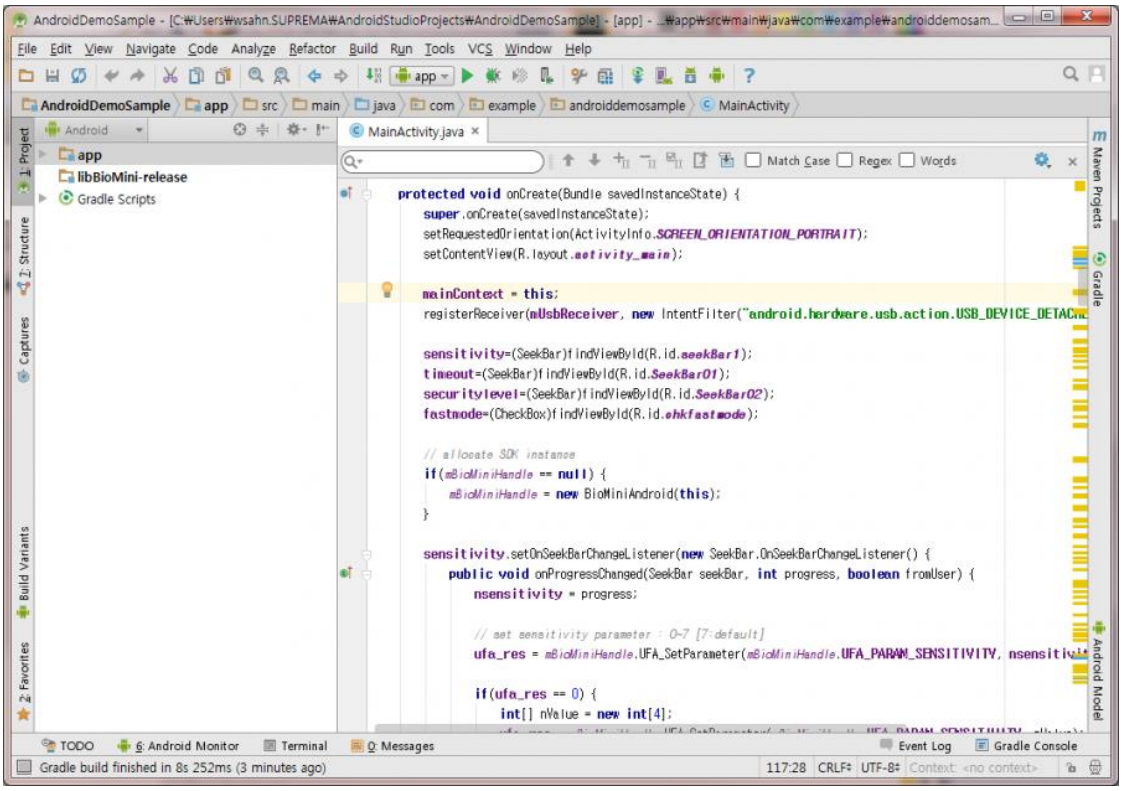
6. Select 'app' or 'other main module' in the left list and click 'Dependencies' tab.



7. Click right mouse button, select '+' and click '3 Module dependency'. After then select :libBioMini_v20.



8. Generate BioMiniAndroid object to control device according to Android SDK programming guide.



3. Development

3.1 Enrollment Tutorial

1. Preliminaries

```
//import biomini android package
import com.suprema.BioMiniFactory;
import com.suprema.CaptureResponder;
import com.suprema.IBioMiniDevice;
```

2. Find scanner device, get user permission and SDK Initialize

```
private static BioMiniFactory mBioMiniFactory = null;

@Override
protected void onCreate(Bundle savedInstanceState) {
.
.
.
// allocate SDK instance
    mBioMiniFactory = new BioMiniFactory(mainContext) {
        @Override
        public void onDeviceChange(DeviceChangeEvent event, Object dev) {

            if (event == DeviceChangeEvent.DEVICE_ATTACHED && mCurrentDevice == null) {
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                    .
                    .
                    .
                        if (mBioMiniFactory != null) {
                            // find BioMini device, get user permission and SDK initialize
                            mCurrentDevice = mBioMiniFactory.getDevice(0);
                        }
                    }
                }
            }).start();
        }
```

3. Parameters

```
// set security level parameter : 1~7 [4:default]
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SECURITY_LEVEL, security_level));
// set sensitivity parameter : 0~7 [7:default]
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SENSITIVITY,
sensitivity_level));
// set timeout parameter : 0~ [10000:default]
```

```
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.TIMEOUT,
timeout));
// set LFD level parameter : 0 ~ 5 [0:default]
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.DETECT_FAKE,
lfd_level));
// set fast mode parameter : 1 or 0 [1:default]
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.FAST_MODE,
fast_mode?1:0));
// set scanning mode parameter : 1 or 0 [1:default]
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SCANNING_MODE, crop_mode?1:0));
// set external trigger parameter : 1 or 0 [1:default]
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.EXT_TRIGGER,
ext_trigger?1:0));
// set auto sleep mode parameter : 1 or 0 [0:default]
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.ENABLE_AUTOSLEEP, auto_sleep?1:0));
```

4. Capture image and extract template

```
((ImageView) findViewById(R.id.imagePreview)).setImageBitmap(null);

IBioMiniDevice.CaptureOption option = new IBioMiniDevice.CaptureOption();
option.captureTemplate = true;

// capture fingerprint image
mCurrentDevice.captureSingle(option,
    new CaptureResponder() {
        @Override
        public boolean onCaptureEx(final Object context, final Bitmap capturedImage,
                                   final IBioMiniDevice.TemplateData capturedTemplate,
                                   final IBioMiniDevice.FingerState fingerState) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(capturedImage != null) {
                        ImageView iv = (ImageView) findViewById(R.id.imagePreview);
                        if(iv != null) {
                            iv.setImageBitmap(capturedImage);
                        }
                    }
                }
            });
            if(capturedTemplate != null) {
                mUsers.add(new UserData(userName, capturedTemplate.data,
                capturedTemplate.data.length));
            }
```

5. Uninitialize scanner module

```
else if (mCurrentDevice != null && event == DeviceChangeEvent.DEVICE_DETACHED &&
mCurrentDevice.isEqual(dev)) {
    mCurrentDevice = null;
}
```

3.2 Verification Tutorial

1. Preliminaries

```
//import biomini android package
import com.suprema.BioMiniFactory;
import com.suprema.CaptureResponder;
import com.suprema.IBioMiniDevice;
```

2. Create matcher

```
private static BioMiniFactory mBioMiniFactory = null;

@Override
protected void onCreate(Bundle savedInstanceState) {
.
.
.
// allocate SDK instance
    mBioMiniFactory = new BioMiniFactory(mainContext) {
        @Override
        public void onDeviceChange(DeviceChangeEvent event, Object dev) {

            if (event == DeviceChangeEvent.DEVICE_ATTACHED && mCurrentDevice == null) {
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                    .
                    .
                    .
                        if (mBioMiniFactory != null) {
                            // find BioMini device, get user permission and SDK initialize
                            mCurrentDevice = mBioMiniFactory.getDevice(0);
                        }
                    }
                }
            }).start();
        }
```

3. Set Parameters

```
// set security level parameter : 1~7 [4:default]
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SECURITY_LEVEL, security_level));
// set sensitivity parameter : 0~7 [7:default]
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SENSITIVITY,
sensitivity_level));
// set timeout parameter : 0~ [10000:default]
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.TIMEOUT,
timeout));
// set LFD level parameter : 0 ~ 5 [0:default]
```

```
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.DETECT_FAKE,
lfd_level));
// set fast mode parameter : 1 or 0 [1:default]
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.FAST_MODE,
fast_mode?1:0));
// set scanning mode parameter : 1 or 0 [1:default]
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SCANNING_MODE, crop_mode?1:0));
// set external trigger parameter : 1 or 0 [1:default]
mCurrentDevice.setParameter(new IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.EXT_TRIGGER,
ext_trigger?1:0));
// set auto sleep mode parameter : 1 or 0 [0:default]
CurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.ENABLE_AUTOSLEEP, auto_sleep?1:0));
```

4. Verify

```
IBioMiniDevice.CaptureOption option = new IBioMiniDevice.CaptureOption();
option.captureTemplate = true;
// capture fingerprint image
mCurrentDevice.captureSingle(option,
    new CaptureResponder() {
        @Override
        public boolean onCaptureEx(final Object context, final Bitmap capturedImage,
                                   final IBioMiniDevice.TemplateData capturedTemplate,
                                   final IBioMiniDevice.FingerState fingerState) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(capturedImage != null) {
                        ImageView iv = (ImageView) findViewById(R.id.imagePreview);
                        if(iv != null) {
                            iv.setImageBitmap(capturedImage);
                        }
                    }
                }
            });
            if(capturedTemplate != null) {
                boolean isMatched = false;
                String matchedName = "";
                for(UserData ud : mUsers) {
                    if(mCurrentDevice.verify(
                        capturedTemplate.data, capturedTemplate.data.length,
                        ud.template, ud.template.length)) {
                        isMatched = true;
                        matchedName = ud.name;
                        break;
                    }
                }
                if(isMatched) {
                    // Match found, Verify OK
                }
                else {
                    // No match found, Verify template not match
                }
            }
```

```
            else {
                // Template is not extracted
            }
            return true;
        }
```

5. Uninitialize scanner module

```
else if (mCurrentDevice != null && event == DeviceChangeEvent.DEVICE_DETACHED &&
mCurrentDevice.isEqual(dev)) {
    mCurrentDevice = null;
}
```

3.3 APIs

1. **getParameter()**

   - Gets the parameter value.

2. **setParameter()**

   - Sets the parameter value.

3. **captureSingle()**

   - Captures single image. Captured image is returned by parameter.

4. **startCapturing()**

   - Starts capturing.

5. **isCaptuing()**

   - Checks whether the scanner is capturing or not.

6. **getFeatureNumber()**

   - Gets the number of Minutiae from template data.

7. **getFPQuality()**

   - Calculates the quality score of an image as defined in NISTIR 7151: FingerPrint Image Quality. The score would be between 1(excellent) and 5(poor).

8. **abortCapturing()**

   - Aborts capturing which is started by startCapturing().

9. **extractTemplate()**

   - Extracts a template from the stored image buffer which is acquired using captureSingle() or startCapturing().

10. **clearCaptureImageBuffer()**

    - Clears the capture image buffer.

11. **verify()**

    - Compares two extracted templates.

12. **getCompanyID()**

    - Returns the company ID of device.

13. **getCoreCoordinate()**

    - Returns the coordinates of the fingerprint Core in the captured image

14. **errString()**

    - Gets the error string from ErrorCode

15. **isAwake()**

    - Check whether the currently connected device is SleepMode

* Supported Device: BioMini Slim(PID: 0x0407), BioMini Slim 2(PID: 0x0408)

16. popPerformanceLog()

- Gets the processing log

17. getImageWidth()

- Gets width of target image

18. getImageHeight()

- Gets height of target image

19. getCaptureImageAsWsq()

- Makes an WSQ image from captured raw image data, with the specific width and height

20. getCaptureImageAsBmp()

- Make a BMP formatted array from the captured image buffer

21. getDeviceInfo()

- Gets the device information

22. setEncryptionKey()

- Sets the encryption key

23. encrypt()

- Encrypts the data

24. decrypt()

- Decrypts the data

25. getLastError()

- Gets the last ErrorCode

26. getSDKInfo()

- Gets the revision information of SDK

27. getDeviceCount()

- Gets the number of devices

28. getDevice()

- Gets the device

29. captureAuto()

- Unimplemented. This API will be implemented

30. activate()

- This API used for device control however it is unnecessary in application development.

31. deactivate()

- This API used for device control however it is unnecessary in application development.

32. isInUse()

- This API used for device control however it is unnecessary in application development.

33. isEqual()

- This API used for device control however it is unnecessary in application development.

Create BioMiniFactory instance

```
private static BioMiniFactory mBioMiniFactory = null;
public IBioMiniDevice mCurrentDevice = null;
private MainActivity mainContext;
.
.
.
@Override
protected void onCreate(Bundle savedInstanceState) {
.
.
.
     mainContext = this;
.
.
.
}
mBioMiniFactory = new BioMiniFactory(mainContext) {
    @Override
    public void onDeviceChange(DeviceChangeEvent event, Object dev) {
        if (event == DeviceChangeEvent.DEVICE_ATTACHED && mCurrentDevice == null) {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    int cnt = 0;
                    while (mBioMiniFactory == null && cnt < 20) {
                        SystemClock.sleep(1000);
                        cnt++;
                    }
                    if (mBioMiniFactory != null) {
                        mCurrentDevice = mBioMiniFactory.getDevice(0);
                    }
                }
            }).start();
        } else if (mCurrentDevice != null && event == DeviceChangeEvent.DEVICE_DETACHED &&
mCurrentDevice.isEqual(dev)) {
            Log.d(TAG, "mCurrentDevice removed : " + mCurrentDevice);
            mCurrentDevice = null;
        }
    }
};
```

3.3.1 IBioMiniDevice.getParameter

- Gets the parameter value.

```
Public Parameter getParameter(ParameterType type);
```

- Parameter

- **`ParameterType type`** : Parameter type; one of parameters

ParameterType definition	Code	Description	Default value
TIMEOUT	0	Timeout (millisecond unit) (0: infinite)	10000
SENSITIVITY	1	Sensitivity (0 ~ 7); Higher value means more sensitive	7
SCANNING_MODE	2	Result image size of BioMini Plus 2	1
FAST_MODE	3	Fast Mode (0: not use fast mode, 1: use fast mode)	1
SECURITY_LEVEL	4	Level 1 : Below 1% (1e~2)	4
		Level 2 : Below 0.1% (1e~3)	
		Level 3 : Below 0.01% (1e~4)	
		Level 4 : Below 0.001% (1e~5)	
		Level 5 : Below 0.0001% (1e~6)	
		Level 6 : Below 0.00001% (1e~7)	
DETECT_FAKE	5	Use Live Finger Detection (0: not use LFD, 1~5 : use LFD); Higher value means more strong to fake finger	0
AUTO_ROTATE	6	Supports 360-degree matching (0: not use rotate mode, 1: use rotate mode)	0
DETECT_CORE	7	Detect core (0: not use the function to detect the core of a fingerprint, 1: use the function to detect the core of a fingerprint)	0

TEMPLATE_TYPE	8	Template type	0
ENABLE_AUTOSLEEP	9	Checks whether auto-sleep mode is or not (0: off, 1: on) * Supported device: BioMini Slim, BioMini Slim 2	0
EXT_TRIGGER	10	Adaptive Capture Mode	1
INVALID	11	Invalid	

-

- Example

```
// Define
IBioMiniDevice mCurrentDevice = null;
.
. // Create BioMiniFactory instance
  .

// Get security level
int security_level = (int)
mCurrentDevice.getParameter(IBioMiniDevice.ParameterType.SECURITY_LEVEL).value;

// Get sensitivity level
int sensitivity_level = (int)
mCurrentDevice.getParameter(IBioMiniDevice.ParameterType.SENSITIVITY).value;

// Get timeout
int timeout = (int)
mCurrentDevice.getParameter(IBioMiniDevice.ParameterType.TIMEOUT).value;

// Get LFD level
int lfd_level = (int)
mCurrentDevice.getParameter(IBioMiniDevice.ParameterType.DETECT_FAKE).value;

// Get fast mode
boolean fast_mode =
mCurrentDevice.getParameter(IBioMiniDevice.ParameterType.FAST_MODE).value == 1;

// Get crop mode
boolean crop_mode =
mCurrentDevice.getParameter(IBioMiniDevice.ParameterType.SCANNING_MODE).value
== 1;

// Get ext trigger mode
boolean ext_trigger =
mCurrentDevice.getParameter(IBioMiniDevice.ParameterType.EXT_TRIGGER).value ==
1;

// Get auto sleep mode
```

```
boolean auto_sleep =
mCurrentDevice.getParameter(IBioMiniDevice.ParameterType.ENABLE_AUTOSLEEP).valu
e == 1;
```

- Return Value(refer to return values)

3.3.2 IBioMiniDevice.setParameter

- Sets the parameter value.

```
public boolean setParameter(Parameter parameter);
```

- Parameter
  - **Parameter parameter** : Parameter to declare

ParameterType definition	**Code**	**Description**	**Default value**
TIMEOUT	0	Timeout (millisecond unit) (0: infinite)	10000
SENSITIVITY	1	Sensitivity (0 ~ 7); Higher value means more sensitive	7
SCANNING_MODE	2	Result image size of BioMini Plus 2	1
FAST_MODE	3	Fast Mode (0: not use fast mode, 1: use fast mode)	1
SECURITY_LEVEL	4	Level 1 : Below 1% (1e~2)	4
		Level 2 : Below 0.1% (1e~3)	
		Level 3 : Below 0.01% (1e~4)	
		Level 4 : Below 0.001% (1e~5)	
		Level 5 : Below 0.0001% (1e~6)	
		Level 6 : Below 0.00001% (1e~7)	
DETECT_FAKE	5	Use Live Finger Detection (0: not use LFD, 1~3 : use LFD); Higher value means more strong to fake finger	0
AUTO_ROTATE	6	Supports 360-degree matching (0: not use rotate mode, 1: use rotate mode)	0
DETECT_CORE	7	Detect core (0: not use the function to detect the core of a fingerprint, 1: use the function to detect the core of a fingerprint)	0

TEMPLATE_TYPE	8	Template type	0
ENABLE_AUTOSLEEP	9	Checks whether auto-sleep mode is or not (0: off, 1: on) * Supported device: BioMini Slim, BioMini Slim 2	0
EXT_TRIGGER	10	Adaptive Capture Mode	1
INVALID	11	Invalid	

-

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;
.
. // Create BioMiniFactory instance.
  .

// Set security level
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SECURITY_LEVEL,
security_level));

// Set sensitivity level
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SENSITIVITY_LEVEL,
security_level));

// Set timeout
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.TIMEOUT, timeout));

// Set LFD level
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.DETECT_FAKE,
lfd_level));

// Set fast mode
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.FAST_MODE,
fast_mode?1:0));

// Set crop mode
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.SCANNING_MODE,
crop_mode?1:0));

// Set ext trigger mode
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.EXT_TRIGGER,
ext_trigger?1:0));
```

```
// Set auto sleep mode
mCurrentDevice.setParameter(new
IBioMiniDevice.Parameter(IBioMiniDevice.ParameterType.ENABLE_AUTOSLEEP,
auto_sleep?1:0));
```

- Return Value(refer to return values)

3.3.3 IBioMiniDevice.captureSingle

- Captures single image. The image and template are passed through the parameters of the Capture responder callback function.

```
boolean captureSingle(CaptureOption opt, ICaptureResponder responder, boolean
 bAsync);
```

- Parameter

- CaptureOption opt : captureSingle Options to use when captureSingle
- ICaptureResponder responder : callback instance of responder receives capture result and error code
- boolean bAsync : asynchronous parameter of capture operation (TRUE: asynchronous / FALSE: synchronous)

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  . // Create BioMiniFactory instance
  .

Private IBioMiniDevice.CaptureOption mCaptureOptionDefault = new IBioMiniDevice.CaptureOption();
Private CaptureResponder mCaptureResponseDefault = new CaptureResponder() {
  @Override
  public boolean onCaptureEx(final Object context, final Bitmap capturedImage, final
IBioMiniDevice.TemplateData capturedTemplate, final IBioMiniDevice.FingerState fingerState) {
  .
    . // Capture success
    .
    return true;
  }
};

  .
  .
  .

// Capture single image
mCurrentDevice.captureSingle(mCaptureOptionDefault, mCaptureResponseDefault, true);
```

- Return Value

true : Capture success

false : Capture fail

3.3.4 IBioMiniDevice.startCapturing

- Starts capturing. The image and template are passed through the parameters of the Capture responder callback function.

```
int startCapturing(CapturOption opt, ICaptureResponder responder);
```

- Parameter
  - CaptureOption opt : options to use when startCapturing

  - ICaptureResponder responder : callback instance of responder receives capture result and error code

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

Private IBioMiniDevice.CaptureOption mCaptureOptionDefault = new IBioMiniDevice.CaptureOption();
Private CaptureResponder mCaptureResponsePrev = new CaptureResponder() {
  @Override
  public boolean onCaptureEx(final Object context, final Bitmap capturedImage, final
IBioMiniDevice.TemplateData capturedTemplate, final IBioMiniDevice.FingerState fingerState) {
  .
    . // Capture success
    .
return true;
  }
};

  .
  .
  .

// Start capturing
mCurrentDevice.startCapturing(mCaptureOptionDefault, mCaptureResponsePrev);
```

- Return Value

  ErrorCode

3.3.5 IBioMiniDevice.isCapturing

- Checks whether the scanner is capturing or not.

```
boolean isCapturing();
```

- Parameter

  - **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Check whether the scanner is capturing
if(!mCurrentDevice.isCapturing()) {
  .
  . // The scanner is not capturing
  .
}
```

- Return Value

  true : if capturing is on going

  false : if not

3.3.6 IBioMiniDevice.getFeatureNumber

- Gets the number of Minutiae from template data

```
int getFeatureNumber(byte[] template, int template_size);
```

- Parameter
  - **byte[] template** : Array of template data
  - **int template_size** : Template size
- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

byte[] template;
int size;

// Capture a fingerprint image & extract template
// Get template data; template

// Get feature number
int nFeature = mCurrentDevice.getFeatureNumber(template, size);
```

- Return Value

Number of features from the template given

3.3.7 IBioMiniDevice.getFPQuality

- Calculates the quality score of an image as defined in NISTIR 7151: FingerPrint Image Quality. The score would be between 1(excellent) and 5(poor)

```
int getFPQuality(byte[]FPImage,int nWidth, int nHeight, int nFPQualityMode);
```

- Parameter

  - byte [] FPImage : fingerprint image in RAW format

  - int nWidth : width of the image

  - int nHeight : height of the image

  - int nFPQualityMode : fingerprint calculation mode

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

byte[] image;
int width;
int height;
int mode;


// Capture a fingerprint image & extract template
// Get captured image data; image, width, height



int nQuality = mCurrentDevice.getFPQuality(image, width, height, mode);
```

- Return Value

  Fingerprint quality

3.3.8 IBioMiniDevice.abortCapturing

- Aborts capturing which is started by startCapturing() or captureSingle()

```
int abortCapturing();
```

- Parameter
  - **N/A**
- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Abort capturing
if(mCurrentDevice.isCapturing()) {
  .
  . mCurrentDevice.abortCapturing();
  .
}
```

- Return Value

  ErrorCode

3.3.9 IBioMiniDevice.extractTemplate

- Extracts a template from the stored image buffer which is acquired using captureSingle() or startCapturing()

```
TemplateData extractTemplate();
```

- Parameter

  - **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Capture a fingerprint image

// Extract template
IBioMiniDevice.TemplateData tmp = mCurrentDevice.extractTemplate();
```

- Return Value

  Template data

3.3.10 IBioMiniDevice.clearCaptureImageBuffer

- Clears the capture image buffer

```
boolean clearCaptureImageBuffer();
```

- Parameter

  - **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Capture a fingerprint image

// Clear capture image buffer
mCurrentDevice.clearCaptureImageBuffer()
```

- Return Value

  true : if image buffer was cleared

  false : if not

3.3.11 IBioMiniDevice.verify

- Compares two extracted templates

```
boolean verify(byte[] pTemplate1, int nTemplate1Size, byte[] pTemplate2, int
nTemplate2Size);
```

- Parameter

  - byte [] pTemplate1 : an array containing the first template

  - int nTemplate1Size : size of the first template array

  - byte [] pTemplate2 : an array containing the second template

  - int nTemplate2Size : size of the second template array

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

byte[] template1;
byte[] template2;
// Capture two fingerprint images and extract templates

// Capture a fingerprint image & extract template
// Get extracted template; template1, template2

if(mCurrentDevice.verify(template1.data, template1.data.length, template2.data,
template2.data.length)) {
  .
  . // Verify success
  .
}
```

- Return Value

  true : verification is succeed

  false : verification is failed

3.3.12 IBioMiniDevice.getCompanyID

- Returns the company ID of device

```
String getCompanyID();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Get company ID
Log.d( “ CompanyID : “ + mCurrentDevice.getCompanyID());
```

- Return Value

Company ID

3.3.13 IBioMiniDevice.getCoreCoordinate

- Returns the coordinates of the fingerprint Core in the captured image

```
int[] getCoreCoordinate();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Capture fingerprint image

// Get coordinates of the fingerprint core
int[] coord = mCurrentDevice.getCoreCoordinate();
```

- Return Value

Core coordinate array

3.3.14 IBioMiniDevice.errString

- Gets the error string from ErrorCode

```
String errString(int errCode);
```

- Parameter

- **int errCode** : Error code in int format

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Any process

// Gets the error string
Log.d( “ errString : “ + mCurrentDevice.errString(errCode));
```

- Return Value

Error string

3.3.15 IBioMiniDevice.isAwake

- Check whether the currently connected device is SleepMode

  * Supported Device: BioMini Slim(PID: 0x0407), BioMini Slim 2(PID: 0x0408)

```
boolean isAwake();
```

- Parameter

  - **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

if(mCurrentDevice.isAwake() {
  .
  . // Wake up mode
  .
}
```

- Return Value

  True : wake up mode

  False : sleep mode

3.3.16 IBioMiniDevice.popPerformanceLog

- Gets the processing log

```
String popPerformanceLog();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .
// Any process

// Gets the processing log
Log.d(((IBioMiniDevice)context).popPerformanceLog());
```

- Return Value

Processing log

3.3.17 IBioMiniDevice.getImageWidth

- Gets width of target image

```
int getImageWidth();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Gets width of target image
int nWidth = mCurrentDevice.getImageWidth();
```

- Return Value

Width of target image

3.3.18 IBioMiniDevice.getImageHeight

- Gets height of target image

```
int getImageHeight();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Gets height of target image
int nHeight = mCurrentDevice.getImageHeight();
```

- Return Value

Height of target image

3.3.19 IBioMiniDevice.getCaptureImageAsWsq

- Makes an WSQ image from captured raw image data, with the specific width and height

```
byte[] getCaptureImageAsWsq(int width, int height, float fBitRate, int rotate);
```

- Parameter

  - int width : width of image to crop (padding if bigger than image size, original size if negative)

  - int height : height of image to crop (padding if bigger than image size, original size if negative)

  - float fBitRate : compression ratio (0 to 8)

  - int rotate : rotation angle of the resulting image

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .
// Capture fingerprint image

int width;
int height;
float fBitRate;
int rotate;

// Get capture image as wsq
byte[] wsq =mCurrentDevice.getCaptureImageAsWsq(width, height, fBitRate,
rotate);
```

- Return Value

  WSQ image buffer

3.3.20 IBioMiniDevice.getCaptureImageAsBmp

- Make a BMP formatted array from the captured image buffer

```
byte[] getCaptureImageAsBmp();
```

- Parameter

  - **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .
// Capture fingerprint image

// Get capture image as bmp
byte[] bmp = mCurrentDevice.getCaptureImageAsBmp();
```

- Return Value

  BMP formatted array

3.3.21 IBioMiniDevice.getDeviceInfo

- Gets the device information

```
DeviceInfo getDeviceInfo();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Get device information
if(mCurrentDevice != null) {
  Log.d(“ DeviceName : “ + mCurrentDevice.getDeviceInfo().deviceName);
}
```

- Return Value

Device information

3.3.22 IBioMiniDevice.setEncryptionKey

- Sets the encryption key

```
void setEncryptionKey(byte[] key);
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .
byte[] encryptionKey;

// Set encryption key
mCurrentDevice.setEncryptionKey(encryptionKey);
```

- Return Value

  null

3.3.23 IBioMiniDevice.encrypt

- Encrypts the data

```
byte[] encrypt(byte[] data);
```

- Parameter

- **byte[] data** : data to be decrypted

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Capture fingerprint image and extract template
// Get extracted template; template

// Encrypt template
byte[] encryptData = mCurrentDevice.encrypt(template.data);
```

- Return Value

Encrypted data

3.3.24 IBioMiniDevice.decrypt

- Decrypts the data

```
byte[] decrypt(byte[] data);
```

- Parameter

- **byte[] data** : data to be encrypted

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Capture fingerprint image and extract template
// Encrypt extracted template data
// Get encrypted template data; data

// Dncrypt template
byte[] decryptData = mCurrentDevice.decrypt(data);
```

- Return Value

Decrypted data

3.3.25 IBioMiniDevice.getLastError

- Gets the last ErrorCode

```
ErrorCode getLastError();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

  .
  .// Create BioMiniFactory instance
  .

// Get last error code
ErrorCode lastError = mCurrentDevice.getLastError();
```

- Return Value

ErrorCode.OK

3.3.26 BioMiniFactory.getSDKInfo

- Gets the revision information of SDK

```
SDKInfo getSDKInfo();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

private static BioMiniFactory mBioMiniFactory=null;

  .
  .// Create BioMiniFactory instance
  .

// Get SDK information
printRev(“”+mBioMiniFactory.getSDKInfo());
```

- Return Value

Revision information of SDK

3.3.27 BioMiniFactory.getDeviceCount

- Gets the number of devices

```
int getDeviceCount();
```

- Parameter

- **N/A**

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

private static BioMiniFactory mBioMiniFactory=null;

  .
  .// Create BioMiniFactory instance
  .

// Get device count
int count = mBioMiniFactory.getDeviceCount();
```

- Return Value

The number of devices

3.3.28 BioMiniFactory.getDevice

- Gets the device

```
IBioMiniDevice getDevice(int i);
```

- Parameter

- **int i** : device number to get

- Example

```
// Define
IBioMiniDevice mCurrentDevice=null;

private static BioMiniFactorym BioMiniFactory=null;

  .
  .// Create BioMiniFactory instance
  .

// Get device
mCurrentDevice = mBioMiniFactory.getDevice(0);
```

- Return Value

Device

3.3.29 IBioMiniDevice.captureAuto

- Unimplemented. This API will be implemented???

3.3.30 IBioMiniDevice.activate

- This API used for device control however it is unnecessary in application development.

3.3.31 IBioMiniDevice.deactivate

- This API used for device control however it is unnecessary in application development.

3.3.32 IBioMiniDevice.isInUse

- This API used for device control however it is unnecessary in application development.

3.3.33 IBioMiniDevice.isEqual

- This API used for device control however it is unnecessary in application development.

3.4 Definitions

3.4.1 ENUM

ImageType definition	Code	Description
RAW_8	0	8-bit RAW image
BITMAP	1	8-bit bitmap image
WSQ	2	WSQ image

DisconnectionCause definition	Code	Description
USB_UNPLUGGED	0	USB Disconnected
SLEEP_MODE	1	SLEEP MODE status
DEACTIVATED	2	Deactivated state

FrameRate definition	Code	Description
LOW	0	Low speed frame rate mode
MID	1	Medium speed frame rate mode
HIGH	2	High speed frame rate mode
ELOW	3	Extreme low speed frame rate mode
SHIGH	4	Superior high speed frame rate mode
SLOW	5	Superior low speed frame rate mode
DEFAULT	6	Default value : MID

ParameterValueType definition	Code	Description
INT	0	Integer

STRING	1	String

ParameterType definition	Code	Description	Default value
TIMEOUT	0	Timeout (millisecond unit) (0: infinite)	10000
SENSITIVITY	1	Sensitivity (0 ~ 7); Higher value means more sensitive	7
SCANNING_MODE	2	Result image size of BioMini Plus 2	1
FAST_MODE	3	Fast Mode (0: not use fast mode, 1: use fast mode)	1
SECURITY_LEVEL	4	Level 1 : Below 1% (1e~2)	4
		Level 2 : Below 0.1% (1e~3)	
		Level 3 : Below 0.01% (1e~4)	
		Level 4 : Below 0.001% (1e~5)	
		Level 5 : Below 0.0001% (1e~6)	
		Level 6 : Below 0.00001% (1e~7)	
DETECT_FAKE	5	Use Live Finger Detection (0: not use LFD, 1~3 : use LFD); Higher value means more strong to fake finger	0
AUTO_ROTATE	6	Supports 360-degree matching (0: not use rotate mode, 1: use rotate mode)	0
DETECT_CORE	7	Detect core (0: not use the function to detect the core of a fingerprint, 1: use the function to detect the core of a fingerprint)	0
TEMPLATE_TYPE	8	Template type	0
ENABLE_AUTOSLEEP	9	Checks whether auto-sleep mode is or not (0: off, 1: on) * Supported device: BioMini Slim, BioMini	0

		Slim 2	
EXT_TRIGGER	10	Adaptive Capture Mode	1
INVALID	11	Invalid	

TemplateType definition	Code	Description
SUPREMA	0	Suprema template type
ISO19794_2	1	ISO template type
ANSI1378	2	ANSI378 template type

ErrorCode definition	Code	Description
OK	0	Success
ERROR	1	General error
ERR_LICENSE_NOT_MATCH	2	License does not match
ERR_NOT_SUPPORTED	3	This function is not supported
ERR_INVALID_PARAMETERS	4	Input parameters are invalid
ERR_ALREADY_INITIALIZED	5	Module is already initialized
ERR_NOT_INITIALIZED	6	Module is not initialized
ERR_NO_DEVICE	7	Device is not connected
ERR_PERMISSION_DENIED	8	Device permission is canceled
ERR_CAPTURE_RUNNING	9	Capturing is started using captureSingle or startCapturing
ERR_CAPTURE_FAILED	10	Capturing is timeout or aborted
ERR_NOT_CAPTURED	11	There is no captured image for extraction
ERR_EXTRACTION_FAILED	12	Extraction is failed

ERR_TEMPLATE_TYPE	13	Template type is not matched
ERR_FILE_EXIST_ALREADY	14	A file with same name exists
ERR_CORE_NOT_DETECTED	15	Core is not detected
ERR_CORE_TO_LEFT	16	Move finger to left
ERR_CORE_TO_LEFT_TOP	17	Move finger to left-top
ERR_CORE_TO_TOP	18	Move finger to top
ERR_CORE_TO_RIGHT_UP	19	Move finger to right-top
ERR_CORE_TO_RIGHT	20	Move finger to right
ERR_CORE_TO_RIGHT_BOTTOM	21	Move finger to right-bottom
ERR_CORE_TO_BOTTM	22	Move finger to bottom
ERR_CORE_TO_LEFT_BOTTOM	23	Move finger to left-bottom
ERR_UNKNOWN	24	Unknown Error

ScanningMode definition	Code	Description
SCANNING_MODE_FULL	0	Result image size of BioMini Plus 2 : 315(W)x354(H) (pixels)
SCANNING_MODE_CROP	1	Result image size of BioMini Plus 2 : 288(W)x340(H) (pixels)

3.4.2 Class

1. CaptureOption

- Contains data of capture option

2. ImageOptions

- Contains data of image option

3. Parameter

- Contains data of parameter

4. TemplateData

- Contains data of template data

5. FingerState

- Contains data of finger state

6. DeviceInfo

- Contains data of device information

7. SDKInfo

- Contains revision information of SDK

3.4.2.1 CaptureOption class

- Contains data of capture option

```
class CaptureOption {
         public int captureTimeout;
         public boolean captureImage;
         public boolean captureTemplate;
         public FrameRate frameRate;
};
```

- Properties

captureTimeout	Controls a timeout for capture
captureImage	Checks whether receive image as a result of capture or not
captureTemplate	Checks whether receive template data as a result of capture or not
frameRate	Controls a frame rate for capture

3.4.2.2 ImageOptions class

- Contains data of image options

```
class ImageOptions {
       public ImageType imageType;
       public float compressionRatio;
};
```

- Properties

imageType	Controls an image type
compressionRatio	Controls a compression ratio of an image

3.4.2.3 Parameter class

- Contains data of parameter

```
class Parameter {
        public ParameterType type;
        public long value;
}
```

- Properties

type	Receives a type of parameter
value	Receives a value of parameter

3.4.2.4 TemplateData class

- Contains data of template data

```
class TemplateData {
        public byte[] data;
        public TemplateType type;
        public int quality;
}
```

- Properties

data	Receives a type of template data
type	Receives a value of template type
quality	Receives a value of template quality

3.4.2.5 FingerState class

- Contains data of finger state

```
class FingerState {
        public boolean isFingerExist;
}
```

- Properties

isFingerExist	Checks whether finger exists or not

3.4.2.6 DeviceInfo class

- Contains data of device information

```
class DeviceInfo {
        public String deviceName;
        public String deviceSN;
        public String versionSDK;
}
```

- Properties

deviceName	Receives a name of device
deviceSN	Receives a serial number of device
versionSDK	Receives a version of SDK

3.4.2.7 SDKInfo class

- Contains revision information of SDK

```
class SDKInfo {
        public intMajor;
        public intMid;
        public int Minor;
        public int Rev;
}
```

- Properties

Major	Receives a major number of SDK version
Mid	Receives a mid number of SDK version
Minor	Receives a minor number of SDK version
Rev	Receives a revision version of SDK

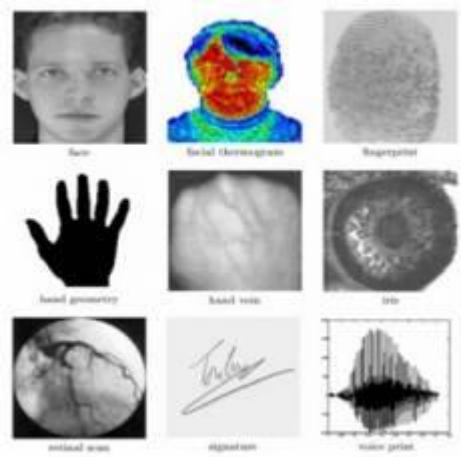
4. Appendix

What is Biometrics?

Identifying individuals based on their distinctive anatomical(fingerprint, face, iris, hand geometry) and behavioral(signature, voice) characteristics is called biometrics. Because biometric identifiers cannot be shared or misplaced, they intrinsically represent an individual's identify. Biometrics is quickly becoming an essential component of effective identification solutions. Recognition of a person by their body, then linking that body to an externally established “identity”, forms a powerful authentication tool.

Fingerprint identification has been widely used for a long time because individual’s unique Fingerprint pattern cannot be shared or misplaced, they essentially represent an individual’s identity. Fingerprint contains rich information in a small area compared to other biological anatomical (face, iris, hand geometry) and behavioral (signature, voice) characteristics. Identifying individuals based on fingerprint is well-known as the most feasible method compared to other biometrical characteristics. Among the other biometric identification methods, fingerprint identification has a good balance of qualities including accuracy, throughput, size and cost of scan devices, maturity of technology and convenience of use, making it the dominant biometric technology in industry.

Two major methods are being used in fingerprint identification; matching based on feature point and filter bank. Matching algorithm based on feature point extract “local minutiae from Thinning fingerprint image or gray scale fingerprint image. Thereafter a method for matching using the location relationship between the feature point in the fingerprint template stored image and the input fingerprint image. he feature point matching takes relatively short processing time however matching performance could be deceased on noised fingerprint images. Filters bank matching algorithm extract fingerprint ridge using Gabor filters bank.



Measured by means of an automated device for physical and behavioral characteristics, Technology to utilize as a means of personal identification

Physical characteristics available - fingerprint, palm print, face, iris veins, etc. Physical and behavioral features available - voice, signature, key tapping, Gait, etc.

Fingerprint is specified patterned from finger’s sweat gland which is consisting of ridge and Valley. Ridge has a certain direction at even interval. Also Ridge (Ridge) has characteristics where lean towards end from certain direction and the ridge (Ridge) part (Junction birurcation) which is divided into two directions.

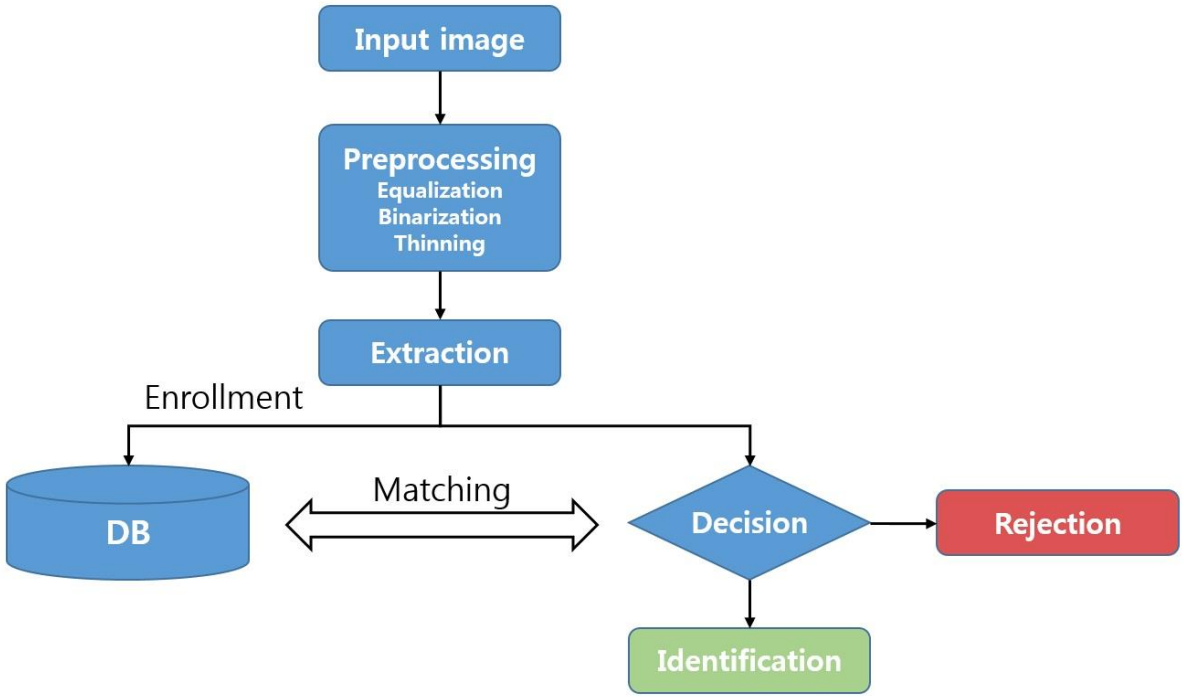
Classfication of fingerprint recoginition

Fingerprint recognition is divided into classification (Classification) and matching (Matching). Classification (Classification) is a way of determine that fingerprint belongs to a particular group based on the overall form of a fingerprint. Matching (Matching) is to compare the fingerprints match with the fingerprints stored in the input. Matching (Matching) is divided into 2 different methods; 1: N matching (Identification) and 1: 1 matching (Verification). The extracted fingerprint minute is used to do matching. The minutiae can be divided into Local feature and global feature

by its own characteristic The Global feature can be determined by formation of fingerprint and direction pattern of ridge & location of ridge minutiae determines the Local feature.

Fingerprint recoginition

Basic formation of fingerprint recognition system is as below. Preprocess the fingerprint image which was entered by fingerprint reader. Then recognize the fingerprint by extracting the features and comparing with saved fingerprint data stored in database.



a. Preprocessing

To extract features from the fingerprint image, follow through below process. Clarifying the image by removing the noise and also ridge ending, minutiae placement, crossed features caused by the noise. This process is called the preprocessing. Here, 3 preprocessing methods are performed; smoothing, binarization and thinning.

b. Feature Extraction

Fingerprint features can be performed as below methods; End point of ridge, minutiae placement, ridge direction pattern, connecting information between core and ridge, local feature containing ridge formation.

The feature extraction can be effected by fingerprint pressure, direction, location, placement, and condition of the fingerprint.

c. Feature Matching

The matching is comparing saved template pattern in the database with fingerprint features taken

from the device. To determine the match, may use pattern matching, statistics identification, structure identification.

About FAR, FRR and EER

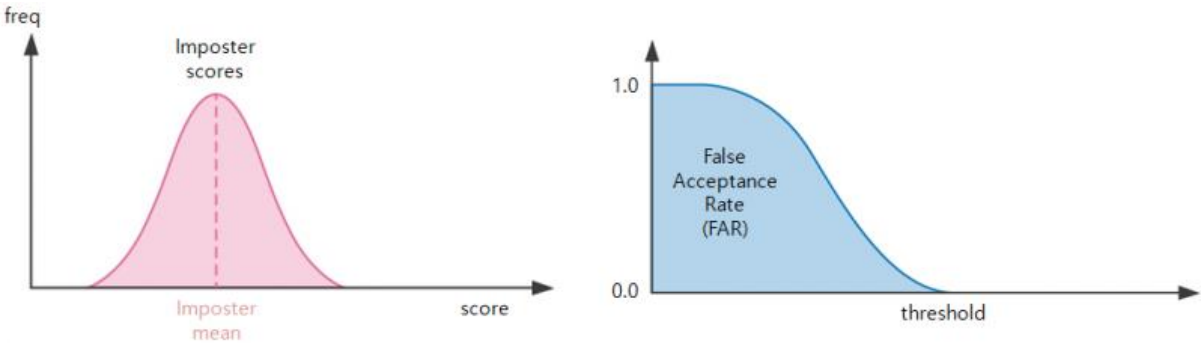
Here, we discusses some general principles of biometric recognition systems, describes different classification errors and explains how the quality of two systems can be compared objectively.

a. Identification vs. Verification

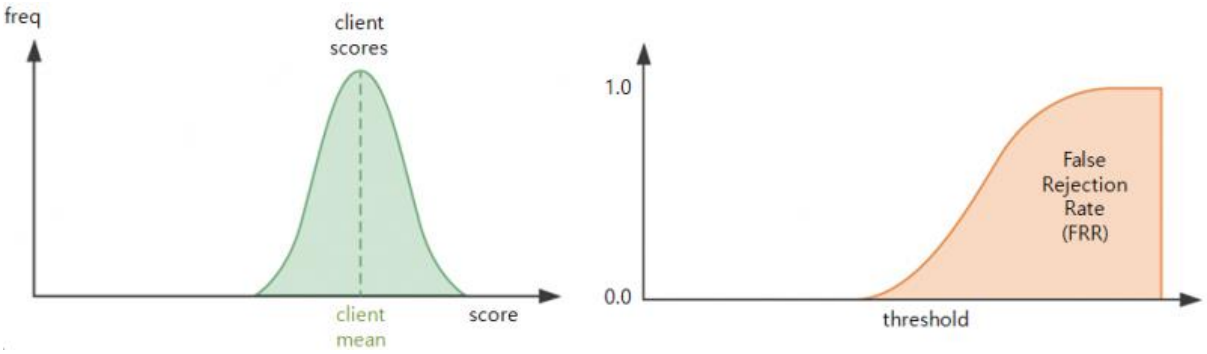
A biometric recognition system can run in two different modes: identification or verification. Identification is the process of trying to find out a person's identity by examining a biometric pattern calculated from the person's biometric features. In the identification case, the system is trained with the patterns of several persons. For each of the persons, a biometric template is calculated in this training stage. A pattern that is going to be identified is matched against every known template, yielding either a score or a distance describing the similarity between the pattern and the template. The system assigns the pattern to the person with the most similar biometric template. To prevent impostor patterns (in this case all patterns of persons not known by the system) from being correctly identified, the similarity has to exceed a certain level. If this level is not reached, the pattern is rejected. In the verification case, a person's identity is claimed a priori. The pattern that is verified only is compared with the person's individual template. Similar to identification, it is checked whether the similarity between pattern and template is sufficient to provide access to the secured system or area.

b. Thresholding (False Acceptance / False Rejection)

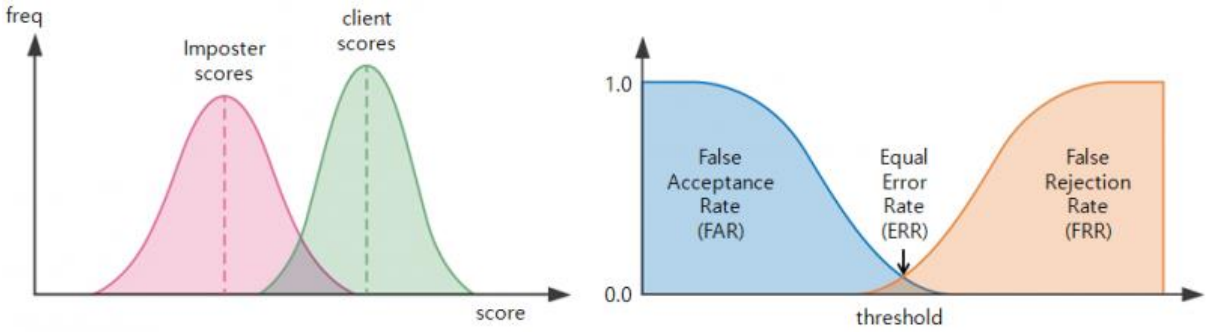
The threshold depending fraction of the falsely accepted patterns divided by the number of all impostor patterns is called **False Acceptance Rate (FAR)**. Its value is one, if all impostor patterns are falsely accepted and zero, if none of the impostor patterns is accepted. Look on the graphic on the right to see the values of the FAR for the score distribution of the left image for varying threshold.



Now let's change to the client patterns. Similar to the impostor scores, the client pattern's scores vary around a certain mean value. The mean score of the client patterns is higher than the mean value of the impostor patterns, as shown in the left of the following two images. If a classification threshold that is too high is applied to the classification scores, some of the client patterns are falsely rejected. Depending on the value of the threshold, between none and all of the client patterns will be falsely rejected. The fraction of the number of rejected client patterns divided by the total number of client patterns is called **False Rejection Rate (FRR)**. According to the FAR, its value lies in between zero and one. The image on the right shows the FAR for a varying threshold for the score distribution shown in the image on the left.



The choice of the threshold value becomes a problem if the distributions of the client and the impostor scores overlap, as shown in the next image on the left. On the right, the corresponding false acceptance and false rejection rates are displayed.



Note that if the score distributions overlap, the FAR and FRR intersect at a certain point. The value of the FAR and the FRR at this point, which is of course the same for both of them, is called the **Equal Error Rate (EER).**

Scanners

BioMini Slim 2

Ultra-slim FAP20 Fingerprint scanner
BioMini Slim 2 is FAP20 certified fingerprint scanner featuring an array of cutting-edge technologies. Along with its 13.5mm slim optical sensor, BioMini Slim 2 features Suprema's proprietary Multi-dynamic Range(MDR) technology, FBI PIV/FIPS 201/Mobile ID FAP20 compliance and Android device support. BioMini Slim 2 also provides developers more physical flexibility with its reduced form factor and the ultra-slim optical sensor ensures robust operation over time. BioMini Slim provides new and advanced Live Finger Detection(LFD) technology by applying a machine learning method that analyzes and categorizes image patterns according to optical characteristics.

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Sensing Area(15.24×20.32mm) ■ Image Size(300×400 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4) ■ IP Rating(IP65)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification(CE, FCC, KC, UL/CB, WHQL, IEC62471, WEEE) ■ Dimensions(72.8×40.7×18.5mm/WxLxH)
Compatibility	■ Operating System(Windows, Linux, Android4.1/Jelly Bean and Above)

BioMini Plus 2

FBI PIV Approved Authentication Scanner

BioMini Plus 2 is an USB fingerprint scanner designed to provide high level security solution for identity access management solutions. BioMini Plus 2 features Suprema's latest live finger detection technology for protection against fake fingerprints. The device features sleek and ergonomic design with scratch resistant IP65 sensor surface securing high usability. The usability is further extended with integration of MDR technology that allows effective capture of fingerprints even under direct sunlight. Combined with its comprehensive SDK solution, BioMini Plus 2 is a versatile and optimal platform for system integrators, hardware manufacturers and security companies.

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Sensing Area(16.0x18.0mm) ■ Image Size(315×354 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4) ■ IP Rating(IP65)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification(CE, FCC, KC, RoHS, USB-IF, CB, IEC62471, WEEE, WHQL) ■ Dimensions(66x90x58mm/WxLxH)
Compatibility	■ Operating System(Windows, Linux, Android4.1/Jelly Bean and Above)

BioMini Slim

FBI PIV and Mobile ID FAP20 Certified USB Fingerprint Scanner

BioMini Slim has been designed to provide a high level security solution for identity access management solutions for authentication. With IP65 grade dust and waterproof form factor, BioMini Slim features a sleek ergonomic design with the latest 500dpi slim optical sensor, which boasts a large platen size for easy and reliable fingerprints capturing as well as advanced LFD (Live Finger Detection) technology.

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Platen Size(18×25.4mm) ■ Sensing Area(17x25mm) ■ Image Size(320×480 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4) ■ IP Rating(IP65)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification(CE,FCC,KC,UL,WHQL, USB-IF,WEEE) ■ Dimensions(82×57.7x27mm/WxLxH)
Compatibility	■ Operating System(Windows, Linux, Android4.1/Jelly Bean and Above)

BioMini Combo

Contact Smart Card Reader with USB Fingerprint Scanner

Suprema BioMini Combo has been designed to provide two factor authentication security solutions for authentication purposes. The scanner features Suprema's latest slim optical sensor with large platen size for easier capturing. Smart card reader functionality and advanced Live Finger Detection technology enhances security makes BioMini Combo a secure platform for developers.

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Platen Size(18×25.4mm) ■ Sensing Area(17x25mm) ■ Image Size(320×480 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378)

	■	Image Format(ISO19794-4)
Card Support	■	CardType(ISO 7816/EMV 2000with SAM Slot_optical)
Interface	■	USB(2.0 High-speed)
Hardware	■	Operating Temperature(-10°C ~ 50°C)
	■	Certification(CE,FCC,KC,UL,WHQL, USB-IF,WEEE)
	■	Dimensions(82×57.7x27mm/WxLxH)
Compatibility	■	Operating System(Windows, Linux, Android4.1/Jelly Bean and Above)

BioMini

High Performance Authentication Scanner Our latest range of high performance fingerprint scanners are supported by powerful software development kit (SDK) which features Suprema's award-winning algorithm that ranked 1st in FVC and NIST MINEX tests. Suprema Authentication Scanners offers unrivaled versatile platform for development to leading security companies, system integrators and hardware manufacturers.

Specification

Main	■	Sensor Type(Optical)
	■	Resolution(500dpi/256gray)
	■	Sensing Area(16x18mm)
	■	Image Size(288×320 pixels)
	■	Compression Standards(WSQ)
	■	Template Format(Suprema, ISO19794-2, ANSI-378)
	■	Image Format(ISO19794-4)
Interface	■	USB(2.0 High-speed)
Hardware	■	Operating Temperature(-10°C ~ 50°C)
	■	Certification(CE,FCC,KC,WHQL)
	■	Dimensions(66x90x58mm/WxLxH)
Compatibility	■	Operating System(Windows, Linux, Android4.1/Jelly Bean and Above)