API Development Manual:

AMTPalmLite SDK For Android

API Version: 12.0

Doc Version: 1.0

June 2022

Thank you for choosing our product. Please read the instructions carefully before operation. Follow these instructions to ensure that the product is functioning properly. The images shown in this manual are for illustrative purposes only.

For further details, please visit our Company's website www.armatura.us.

Trademark

ARMATURA is a registered trademark of ARMATURA LLC. Other trademarks involved in this manual are owned by their respective owners.

Disclaimer

This manual contains information on the operation and maintenance of the ARMATURA product. The copyright in all the documents, drawings, etc. in relation to the ARMATURA supplied product vests in and is the property of ARMATURA. The contents hereof should not be used or shared by the receiver with any third party without express written permission of ARMATURA.

The contents of this manual must be read as a whole before starting the operation and maintenance of the supplied product. If any of the content(s) of the manual seems unclear or incomplete, please contact ARMATURA before starting the operation and maintenance of the said product.

It is an essential pre-requisite for the satisfactory operation and maintenance that the operating and maintenance personnel are fully familiar with the design and that the said personnel have received thorough training in operating and maintaining the machine/unit/product. It is further essential for the safe operation of the machine/unit/product that personnel have read, understood, and followed the safety instructions contained in the manual.

In case of any conflict between terms and conditions of this manual and the contract specifications, drawings, instruction sheets or any other contract-related documents, the contract conditions/documents shall prevail. The contract specific conditions/documents shall apply in priority.

ARMATURA offers no warranty, guarantee, or representation regarding the completeness of any information contained in this manual or any of the amendments made thereto. ARMATURA does not extend the warranty of any kind, including, without limitation, any warranty of design, merchantability, or fitness for a particular purpose.

ARMATURA does not assume responsibility for any errors or omissions in the information or documents which are referenced by or linked to this manual. The entire risk as to the results and performance obtained from using the information is assumed by the user.

ARMATURA in no event shall be liable to the user or any third party for any incidental, consequential, indirect, special, or exemplary damages, including, without limitation, loss of business, loss of profits, business interruption, loss of business information or any pecuniary loss, arising out of, in connection with, or relating to the use of the information contained in or referenced by this manual, even if ARMATURA has been advised of the possibility of such damages.

This manual and the information contained therein may include technical, other inaccuracies, or typographical errors. ARMATURA periodically changes the information herein which will be incorporated into new additions/amendments to the manual. ARMATURA reserves the right to add, delete, amend, or modify the information contained in the manual from time to time in the form of circulars, letters, notes, etc. for better operation and safety of the machine/unit/product. The said additions or amendments are meant for improvement /better operations of the machine/unit/product and such amendments shall not give any right to claim any compensation or damages under any circumstances.

ARMATURA shall in no way be responsible (i) in case the machine/unit/product malfunctions due to any non-compliance of the instructions contained in this manual (ii) in case of operation of the machine/unit/product beyond the rate limits (iii) in case of operation of the machine and product in conditions different from the prescribed conditions of the manual.

The product will be updated from time to time without prior notice. The latest operation procedures and relevant documents are available on http://www.armatura.com.

If there is any issue related to the product, please contact us.

ARMATURA Headquarters

Address 190 Bluegrass Valley Pkwy,
Alpharetta, GA 30005, USA.

For business-related queries, please write to us at: info@armatura.us.

To know more about our global branches, visit www.armatura.us.

About the Company

ARMATURA is a leading global developer and supplier of biometric solutions which incorporate the latest advancements in biometric hardware design, algorithm research & software development. ARMATURA holds numerous patents in the field of biometric recognition technologies. Its products are primarily used in business applications which require highly secure, accurate and fast user identification.

ARMATURA biometric hardware and software are incorporated into the product designs of some of the world's leading suppliers of workforce management (WFM) terminals, Point-of-Sale (PoS) terminals, intercoms, electronic safes, metal key lockers, dangerous machinery, and many other products which heavily rely on correctly verifying & authenticating user's identity.

About the Manual

This manual introduces the operations of **AMTPalmLite SDK For Android**.

All figures displayed are for illustration purposes only. Figures in this manual may not be exactly consistent with the actual products.

Document Conventions

Conventions used in this manual are listed below:

GUI Conventions

For Software	
Convention	**Description**
Bold font	Used to identify software interface names e.g. **OK**, **Confirm**, **Cancel**.
>	Multi-level menus are separated by these brackets. For example, File > Create > Folder.
For Device	
Convention	**Description**
< >	Button or key names for devices. For example, press <OK>.
[ ]	Window names, menu items, data table, and field names are inside square brackets. For example, pop up the [New User] window.
/	Multi-level menus are separated by forwarding slashes. For example, [File/Create/Folder].

Symbols

Convention	Description
	This represents a note that needs to pay more attention to.
	The general information which helps in performing the operations faster.
	The information which is significant.
	Care taken to avoid danger or mistakes.
	The statement or event that warns of something or that serves as a cautionary example.

Table of Contents

1 Introduction

This document will provide with basic SDK development guide and technical background to help with better use of AMTPalmLite SDK document. From the perspective of a developer, the key design objective of this SDK is its compatibility and ease of execution.

This development manual contains the product development documentation for developers that describes the functions provided by the SDK and its related usage, which eases the development environment.

The following sections explain all the required information on how to perform and integrate AMTPalmLite SDK.

1.1 Overview of the SDK

The palm has a complex vascular pattern that is unique to every person. Since the vein patterns lie under the skin, they are almost impossible to replicate/spoof and allow for highly secure authentication with false Near-infrared (NIR) light palm recognition employs a particular image capture technology in which the mounted NIR LED light illuminates the palm, the camera captures the infrared light instead of visible light reflected from the illuminated palm and forms grayscale level images.

NIR light can penetrate the palm skin, the palm surface and the subcutaneous tissue have different levels of infrared light absorption, thus the IR camera captures the pattern characteristics from both palm surface print and subcutaneous vessels (or palm veins). Such biometric patterns are unique and stable to the individuals, not changing with age.

NIR light has different wavelengths from visible light, it allows the camera less impacted by the visible light, therefore the technology can be applied to a vast variety of lighting conditions, especially in very poor-lighted environments.

Recently, significant progress has been made on palm recognition algorithms and imaging sensors, and the palm recognition technology provides advanced features such as touchless authentication, recognition from wide range distance, high pose tolerance and less privacy concern comparing to face recognition technology, the palm recognition-based applications have been widely explored and deployed, especially in access control and security industries.

AMT Palm SDK is a wrapper of Armatura near-Infrared light palm recognition algorithm. It is an excellent 3-in-1 combination of Palm, Palm print and Palm Vein near infrared palm recognition algorithm developed for resist complex ambient light, high tolerance of gesture and large capacity recognition. The algorithm focuses on improving the wide adaptation to the user environment and user habits, thereby greatly improving the robustness and pass rate of palm recognition.

The SDK provides the rich interfaces to access the algorithm's functionalities for palm recognition process, including face detection, feature extraction, liveness detection, template creation and palm identification.

The PalmLite SDK utilizes the widely supported libusb API for palm module communication, supports common-used Windows, Android and Linux operation systems, frees the developers from intimidating hardware operations. It is a developer-friendly toolkit to empower the biometric features on the software application with easy pickup.

The simple library components aid in supporting and enhancing the security requirements through biometric palm recognition which avoids spoofing and has been widely used in various systems, including attendance, security, video monitoring and so on.

1.2 Feature of the SDK

- **Adaptable to Various Environments:**

The PalmLite algorithm uses the region-based image process to improve the palm image quality, or it is to find the area covered by the palm on the image then apply image-enhancing algorithm to improve the image quality. This approach prevents the ambient light's interference on the captured image and increases the recognition rate even on blurred palm images.

Compared to visible light imaging approach, the infrared light imaging approach is more robust under various lighting condition, it makes the PalmLite algorithm well adapted to perform palm recognition on the images captured from a broader range of deployment environments.

- **High Tolerant on Palm Posture**

The uniqueness of the PalmLite algorithm is highly tolerant to the palm postures, it can identify the palm in various postures, including the palm in relaxed or tightly tensed postures,

or in wide yaw, pitch or roll angles. The algorithm is highly adaptable to the way the palm device is installed and allows user to scan the palm in a natural and comfort posture in enrollment or identification scenarios.

- **Accurate and Robust Palm Recognition:**

The PalmLite algorithm selects the palm's stable features and breaks into multi-dimensional vector features such as palm print and palm vein spacing, bifurcations, textures, and curvatures for recognition process. Such features are rich in details, long-lasting, distinguishable, and unique to individuals.

During the process to register the palm template, the PalmLite algorithm takes the averaging approach on multiple templates (5 templates consecutively) to build a stable and robust representation of the candidate palm.

The combination of above processes ensures the algorithm to achieve highly accurate and robust recognition performance.

- **Liveness Detection:**

The pattern from live subcutaneous tissues is invisible to human eyes and non-duplicatable, naturally it provides anti-spoofing security, the combination of NIR camera and PalmLite algorithm makes the palm recognition super secure.

- **High Recognition Performance**

The PalmLite algorithm uses a multi-level matching mode to provide a high verification/identification speed while ensuring stable verification/identification effect. The performance tested on the single-core CPU from standard PC can reach 1 million times per second.

- **Algorithm Integrity**:

Combined with Armatura near-infrared light palm module, the PalmLite algorithm ensures the quality of images along with data integrity for genuine and accurate image recognition process.

1.3 Advantage of the SDK

- Easy to use by other developers.
- Thorough documentation to explain how your code works.
- Enough functionality so it adds value to other applications.
- Does not negatively impact.
- Plays well with other SDKs.

2 Technical Specifications

Development Language

This SDK provides a jar package to support Java development.

Platform Requirements

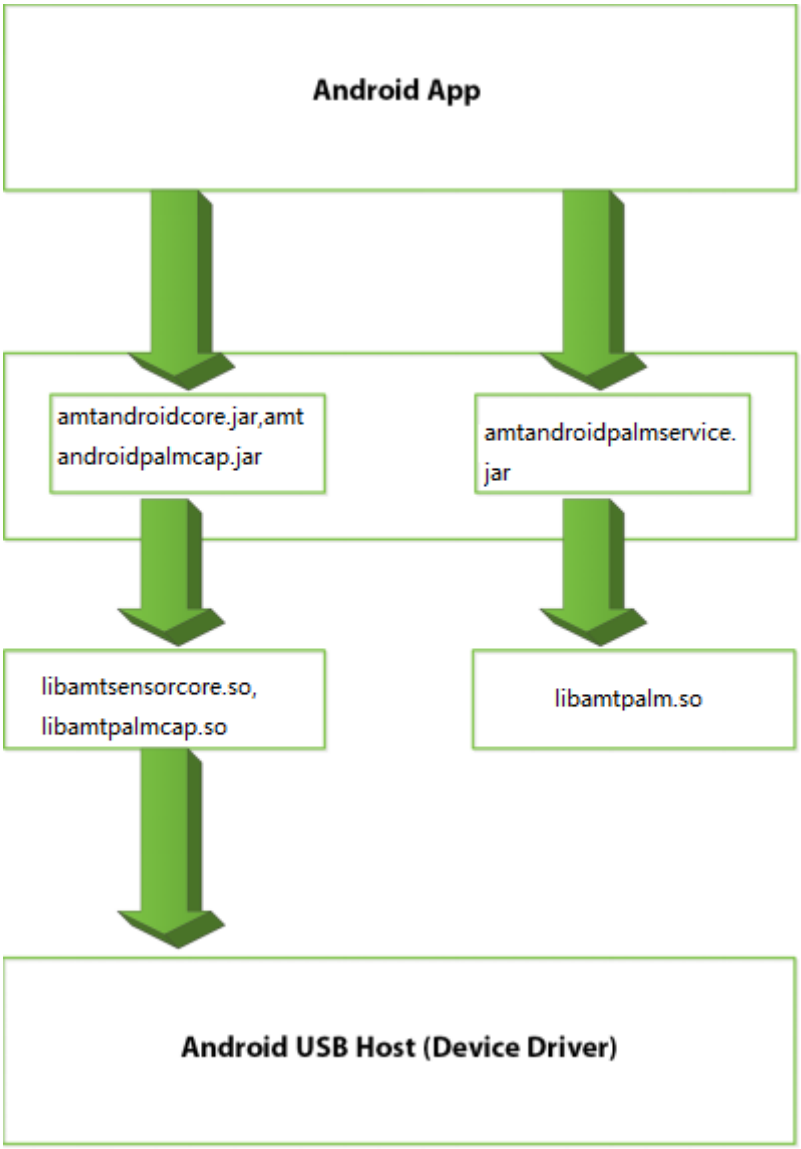
The SDK must be used on Android 4.1 or later, which supports Android USB Host.

Technical Parameters

Parameter	Description
Image resolution	480 x 640
Template size	8848 bytes
Gesture adaptability	Yaw ≤ 20°, Pitch ≤ 20°, Roll ≤ 90°, Bend ≤ 15°
Palm detection	< 50 ms
Palm feature extraction	< 220 ms
Palm verification/identification (1:6000)	< 150 ms
Number of palms supported	6000
Accuracy	TAR = 98.2% when FAR = 0.05%

The preceding algorithm capability indicators are all measured based on an actual image data set (resolution of 480 x 640) and quad-core Cortex-A9, 1.5 GHz processor.

2.1 Architecture



2.1.1 SDK File

- Copy the following files from the libs directory to the app libs directory.
- Add the jniLib dependent library to the android node in the gradle file.

```
sourceSets.main
   {
            jniLibs.srcDir 'libs'
            jni.srcDirs = []
                    }
```

File Name	Description
amtandroidcore.jar	Java library for communication media (such as USB and serial port)
amtandroidpalmcap.jar	Java library for palm capturing
amtandroidpalmservice.jar	Java library for the palm recognition algorithm interface
libamtsensorcore.so	Dynamic link library for underlying communication interfaces of the device
libamtpalmcap.so	Dynamic link library for palm capturing
libamtpalm.so	Dynamic link library for the palm recognition algorithm interface

2.1.2 Project Setup

The dynamic link libraries cannot be compressed and need to be added in the app/build.gradle file.

```
...
android {
    packagingOptions {
            doNotStrip "*/armeabi-v7a/*.so"
            doNotStrip "*/arm64-v8a/*.so"
                }
...
```

2.1.3 USB Information and Permission Configuration

Palm recognition devices

Device Name	Vendor ID	Product ID
AMT-PVM-10	0x34c9	0x2121
AMT-PAR-10	0x34c9	0x2181

USB dongle

- The AMTPalm12.0 algorithm uses a dongle for user authorization. The dongle is usually built in within the palm recognition devices. Therefore, you do not require an external dongle.
- Vendor ID: 0x34c9
- Product ID: 0x0601

Permission configuration: See the demo or "Android USB Host Helper.md."

2.2 Programming Guide

The AMTPalmLite SDK provides two sets of APIs to meet the different requirements of the developers. AMTPalmApi library is recommended.

- When you use the device library (PalmSensor.class) and algorithm library (AMTPalmService12), you need to control the registration process and palm detection process manually.
- If you use AMTPalmApi, the SDK integration will be simpler and faster, because AMTPalmApi encapsulates the PalmSensor and AMTPalmService12 interfaces to implement the registration process and palm detection process automatically.

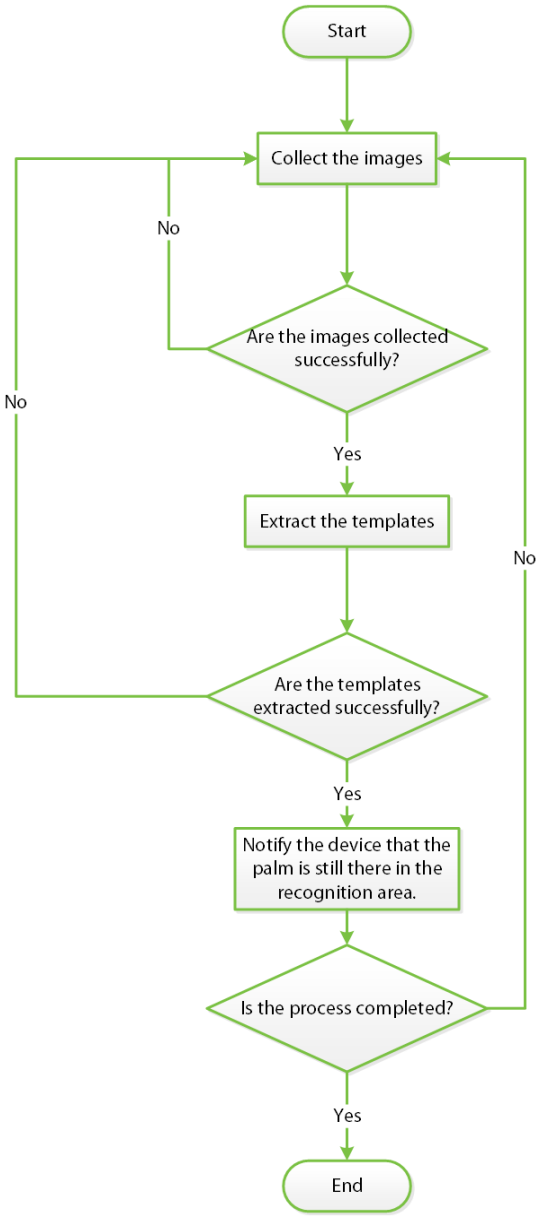
This section describes the key processes of palm recognition to help developers understand the palm registration and detection processes implemented by AMTPalmApi library.

2.2.1 Palm Detection Process

The palm device supports infrared imaging, and the application calls the image capturing function to acquire palm images. When a palm approaches the device, the device turns on the near-infrared fill light, captures palm images, and returns the images. If palm images cannot be captured, the device returns a failure message. After capturing the palm images, the application calls the algorithm library to extract a template. If the template is extracted successfully, the application notifies the device. If the device does not receive the notification within the timeout period (5s by default), it turns off the near-infrared LED light.

Note: Skip this process if you are using AMTPalmApi for SDK integration.

Palm Detection Process Flow



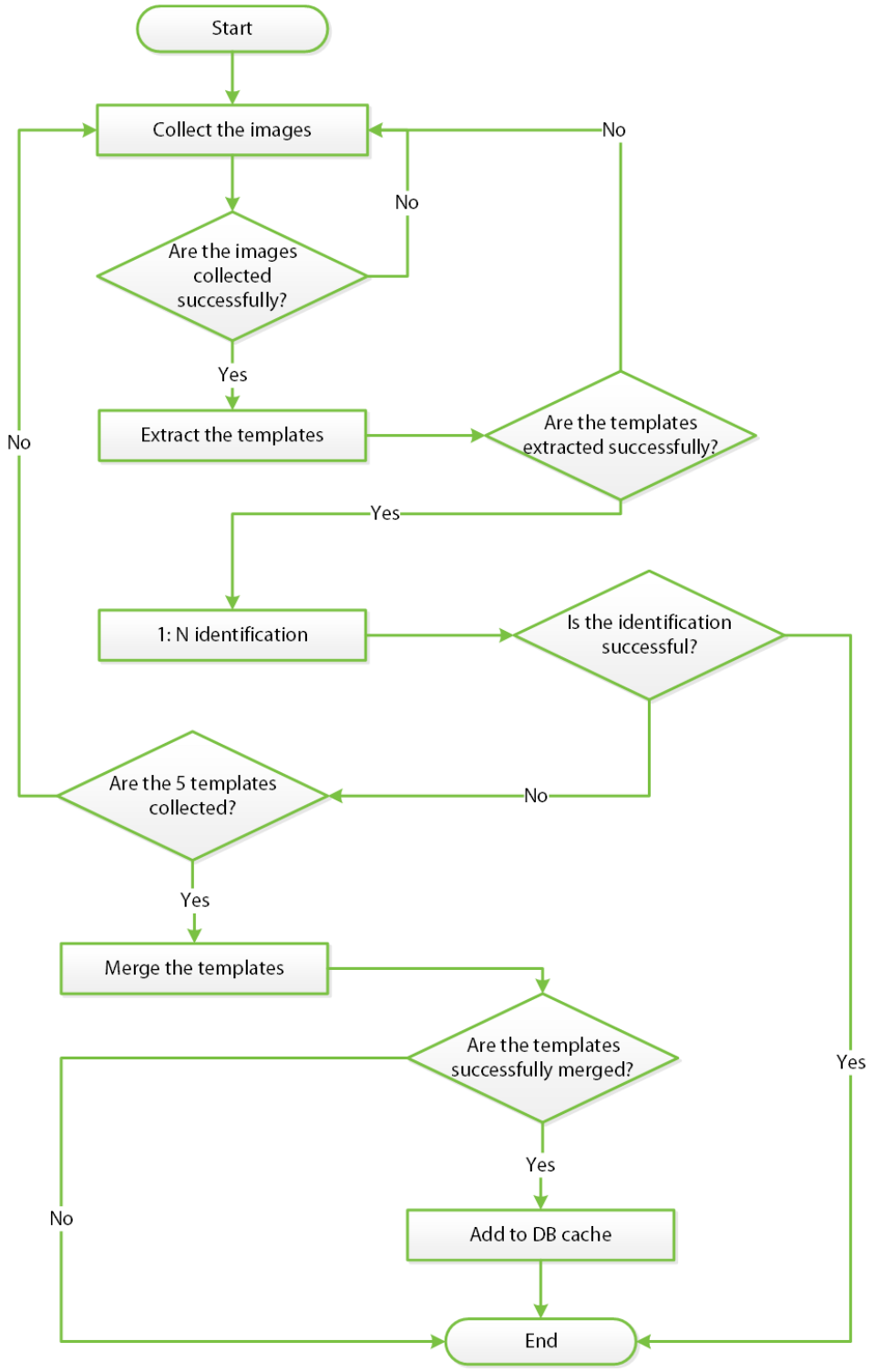
Process Description

- The application calls the startCapture function, so that the SDK starts to capture palm images continuously.
- The SDK uses the callback function onCapture to notify the application about the image capturing result.
- If onCapture returns success, the application calls the extract function to extract the templates.
- After the template is extracted, the application calls palmSensor.setParameter(index, 2010, int2ByteArray(3), 4) to notify the device that the palm is not moved away from the recognition area.

2.2.2 Registration Process

In the palm registration process, the palm device must collect five pre-registered templates and merge them into a registered template. For details about different types of templates, see the interface description.

Registration Process Flow



Process Description

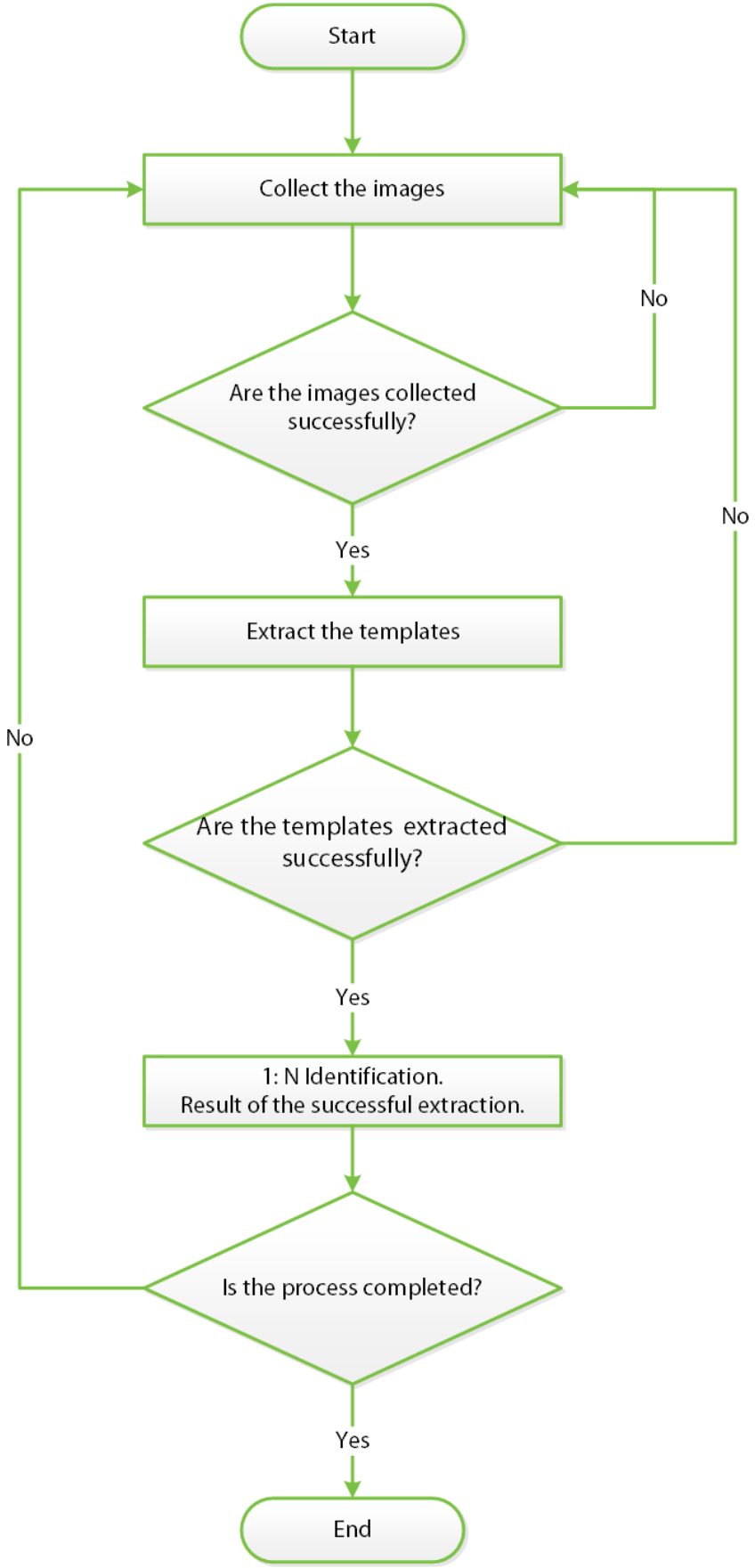
- The application calls the startCapture function, so that the SDK starts to capture palm images continuously.
- The SDK uses the callback function onCapture to notify the application about the image capturing result.
- If onCapture indicates a success, the application calls the extract function to extract the templates.
- The application calls the dbIdentify 1:N function to determine whether the current template has been registered.
- But, if the extracted templated is not registered, the application checks whether five templates have been captured.
- And if less than five templates have been captured, the application continues to capture the next template.
- After capturing five templates, the application merges the templates into a registered template. If the registration fails, the application returns a message and ends the registration process.
- If the registration succeeds, the application calls the dbAdd function to add the registered template to the database.
- And thus, ends the process.

2.2.3 Verification/Identification Process

1:N Identification Process

To implement 1:N palm identification, it is required to add all the registered templates to the database. It is recommended to call the dbAdd function to add all registered templates to the database after successful algorithm initialization.

Identification Process Flow



Process Description

- The application calls the startCapture function, so that the SDK starts to capture palm images continuously.
- The SDK uses the callback function onCapture to notify the application about the image capturing result.
- If onCapture indicates success, the application calls the extract function to extract a template.
- Then the application calls the dbIdentify 1:N function to compare the current template with the registered templates.
- And once the registered template is identified, the application ends the registration process.

3 SDK Interface Description

3.1 Template Format

Template Type	Data Length	Description
Pre-registered template	▪ 99120 bytes ▪ MTPalmService12.FIX_PREREG_TEMPLATE_LEN	Only used for merging into a registered template
Registered template	▪ 8848 bytes ▪ MTPalmService12.FIX_REG_TEMPLATE_LEN	Registered template
Verification/Identification template	▪ 27120 bytes ▪ MTPalmService12.FIX_VER_TEMPLATE_LEN	Only used for palm verification/identification

3.2 Tool Interface

3.2.1 LogHelper.class

This is an algorithm interface class.

Function List

Class/Interface	Description
com.armatura.android.biometric.core.utils.LogHelper.class	Log interface class, used to set the log level
setLevel	Sets the log level

setLevel

Function Syntax

public static void setLevel(int level)

Description

Sets the log level for the SDK.

Parameters

Parameter	Description
level	**In**: Log.VERBOSE to Log.ASSERT.

Remarks

Click here to view the Function List.

3.3 AMTPalmApi Library

3.3.1 AMTPalmApi.class

It is an advanced interface class of the AMTPalmLite SDK, which simplifies the process of calling PalmSensor.class, AMTPalmService.class that helps to complete the SDK integration more easily and quickly.

Function List

Class/Interface	Description
com.armatura.android.biometric.easyapi.AMTPalmApi.class	AMTPalmApi interface class
getSDKVersion	Gets the SDK version
initalize	Connects the device and initializes the algorithm
unInitialize	Releases the algorithm resources and disconnects the device
reset/resetEx	Resets the device
getImageWidth	Gets the image width
getImageHeight	Gets the image height
getSerialNumber	Gets the serial number of the device
getFirmwareVersion	Gets the firmware version
setParameter	Sets parameters
setAMTPalmApiListener	Sets the listener object
setCaptureInterval	Sets the image capturing interval on the backend
startCapture	Starts image capturing
stopCapture	Stops image capturing
startEnroll	Starts registration

cancelEnroll	Cancels registration
controlLED	Controls the status of the LED
verify	Performs 1:1 palm verification
dbAdd	Adds a registered template to the database
dbDel	Removes the specified registered template from the database
dbClear	Clears the database
dbCount	Gets the number of registered templates in the database
dbIdentify	Performs 1:N palm identification
dbVerify	Performs 1:1 palm verification

getSDKVersion

Function Syntax

public static String getSDKVersion()

Description

Gets the SDK version.

Returns

Version number

Remarks

Click here to view the Function List.

initalize

Function Syntax

public int initalize(Context context, int vid, int pid)

Description

Connects the device and initializes the algorithm.

Parameter

Parameter	Description
context	**In:** Application context (getApplicationContext)
vid	**In:** Vendor ID of the palm recognition device
pid	**In**: Product ID of the palm recognition device

Returns

0	Success
Other values	Failure

Remarks

- It is recommended to get the USB permission for the palm recognition devices and the dongle before initializing the algorithm.
- Click here to view the Function List.

unInitialize

Function Syntax

public void unInitialize()

Description

Releases algorithm resource and disconnects the device.

Remarks

- Click here to view the Function List.

reset

Function Syntax

public void reset(int index) throws PalmException

Description

Resets the device.

Parameter

Parameter	Description
index	**In:** Enter 0.

Remarks

- This function can be called only after the initalize function is called successfully.
- Click here to view the Function List.

resetEx

Function Syntax

public void resetEx(int index) throws PalmException

Description

Resets the device.

Parameter

Parameter	Description
index	**In:** Enter 0.

Remarks

- It is not required to call the initalize function before calling this function.
- Click here to view the Function List.

getImageWidth

Function Syntax

public int getImageWidth()

Description

Gets the image width.

Returns

Image width.

Remarks

- Click here to view the Function List.

getImageHeight

Function Syntax

public int getImageHeight()

Description

Gets the image height.

Returns

Image height.

Remarks

- Click here to view the Function List.

getSerialNumber

Function Syntax

public String getSerialNumber()

Description

Gets the serial number of the device.

Returns

Serial number of the device.

Remarks

- Click here to view the Function List.

getFirmwareVersion

Function Syntax

public String getFirmwareVersion()

Description

Gets the firmware version.

Returns

Firmware version.

Remarks

- Click here to view the Function List.

setParameter

Function Syntax

public int setParameter(int code, byte[] paramValue, int size)

Description

Sets parameters.

Parameter

Parameter	Description
code	**In:** Parameter code.
paramValue	**In**: Parameter value
size	**In**: Length of the parameter value

Returns

0	Success
Other values	Failure

Remarks

- Click here to view the Function List.

setAMTPalmApiListener

Function Syntax

public void setAMTPalmApiListener(AMTPalmApiListener listener)

Description

Sets a listener interface object.

Parameter

Parameter	Description
listener	**In**: AMTPalmApiListener interface object

Remarks

- It is recommended to set the listener interface object before calling the startCapture function.
- Click here to view the Function List.

setCaptureInterval

Function Syntax

public void setCaptureInterval(int captureInterval)

Description

Sets the capturing interval

Parameter

Parameter	Description
captureInterval	**In**: Interval time(Unit: millisecond).

Remarks

- Click here to view the Function List.

startCapture

Function Syntax

public boolean startCapture()

Description

Starts image capturing.

Remarks

- After startCapture is called successfully, the SDK captures palm images continuously. After images are captured successfully, the SDK extracts palm features.
- The SDK calls AMTPalmApiListener.onCapture to notify the application about the image capturing result, and calls AMTPalmApiListener.onMatch to notify the application about the feature verification/identification result.
- Click here to view the Function List.

stopCapture

Function Syntax

public void stopCapture()

Description

Stops image capturing.

Remarks

- Click here to view the Function List.

startEnroll

Function Syntax

public void startEnroll()

Description

Starts registration.

Remarks

- Every time when a template is extracted in a registration process, the SDK calls AMTPalmApiListener.onEnroll to notify the application of information about the template extraction, such as the number of images captured, verification/identification template, and registered template.
- Click here to view the Function List.

cancelEnroll

Function Syntax

public void cancelEnroll()

Description

Cancels registration.

Remarks

- Click here to view the Function List.

controlLED

Function Syntax

public boolean controlLED(int ledIndex, int timeout)

Description

Controls status LEDs.

Parameter

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>index</td><td>In: Enter 0</td></tr>
<tr><td>ledIndex</td><td>In:<table><tr><td>0</td><td>Turns off all LEDs</td></tr><tr><td>1</td><td>Turns on the red LED</td></tr><tr><td>2</td><td>Turns on the green LED</td></tr><tr><td>3</td><td>Turns on the blue LED</td></tr></table></td></tr>
<tr><td>timeout</td><td>In: Timeout period after which the device will turn off the status LEDs.
Value range: 0-255 (unit: 100 ms)</td></tr>
</table>

Returns

true	Success
false	Failure

Example

```
// Turn on the red LED and keep it on for 500 ms.
int ledIndex = 1;// Red LED on
int timeout = 5; // Off 500 ms later
palmsensor.controlLED(0, ledIndex, timeout);
```

Remarks

- Click here to view the Function List.

verify

Function Syntax

public int verify(byte[] regTemplate, byte[] verTemplate)

Description

Performs palm verification.

Parameter

Parameter	Description
regTemplate	**In:** A registered template (See mergeRegTemplate.)
verTemplate	**In**: A verification template (See AMTPalmExtractResult.verTemplate and other extract functions.)

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- The score range is 0-1000.
- The recommended threshold value is 576.
- Click here to view the Function List.

dbAdd

Function Syntax

public int dbAdd(String id, byte[] regTemplate)

Description

Adds the registered template to the database.

Parameter

Parameter	Description

id	**In:** Palm ID
regTemplate	**In:** The registered template (See mergeRegTemplate.)

Returns

0	Success
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

dbDel

Function Syntax

public int dbDel(String id)

Description

Removes the template of the specified ID from the database.

Parameter

Parameter	Description
id	**In:** Palm ID

Returns

0	Success
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

dbCount

Function Syntax

public int dbCount()

Description

Gets the total count of the registered templates stored in the database.

Returns

>=0	Number of palm templates
-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

dbClear

Function Syntax

public int dbClear()

Description

Clears the database.

Returns

0	Success
-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

dbIdentify

Function Syntax

public int dbIdentify(byte[] verTemplte, String[] id)

Description

Recognizes a palm.

Parameter

Parameter	Description
verTemplte	**In:** A identification template (See extract functions.)
id	**Out:** ID of the palm recognized, for which the caller needs to allocate memory (exp:id = new String[1]).

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- The score range is 0-1000.
- The recommended threshold value is 576.
- Click here to view the Function List.

dbVerify

Function Syntax

public int dbVerify(byte[] verTemplate, String id)

Description

Performs palm verification.

Parameter

Parameter	Description

verTemplate	**In:** A verification template (See extract functions.)
id	**In:** Palm ID.

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- The score range is 0-1000.
- The recommended threshold value is 576.
- Click here to view the Function List.

3.3.2 AMTPalmApiListener.class

It is a callback listener interface class.

Function List

Class/Interface	Description
com.armatura.android.biometric.easyapi.AMTPalmApiListener.class	Callback listener interface class
onCapture	Callback function that indicates the image capturing result
onException	Callback function that indicates an exception on the device
onMatch	Callback function that indicates the verification/identification extraction result
onEnroll	Callback function that indicates the palm registration status or result
onFeatureInfo	Callback function that indicates the palm feature status

onCapture

Function Syntax

public void onCapture(int actionResult, byte[] palmImage)

Description

Callback function that indicates the image capturing result.

Parameter

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>actionResult</td><td>In: Result of palm image capturing:
<table><tr><td>0</td><td>Indicates that images has been captured successfully and can be displayed.</td></tr><tr><td>Other values</td><td>Indicate the failure to capture palm images and can be ignored</td></tr></table></td></tr>
<tr><td>palmImage</td><td>In:
<table><tr><td>actionResult = 0</td><td>This parameter indicates the gray scale in the Gray256 system.</td></tr><tr><td>actionResult! = 0</td><td>The parameter value is null.</td></tr></table></td></tr>
</table>

Remarks

- After startCapture is called successfully, the SDK calls this callback function to notify the application every time it captures a palm image.
- Click here to view the Function List.

onException

Function Syntax

public void onException()

Description

Callback function that indicates an exception on the device.

Remarks

- When an exception occurs in communication, the SDK calls this callback function repeatedly to notify the application about the exception.
- It is recommended to call the resetEx function when this callback function is triggered for the first time.
- Then, the SDK listens to the USB unplug/plug notification and reconnects the device.

- Click here to view the Function List.

onMatch

Function Syntax

public void onMatch(int actionResult, byte[] verTemplate)

Description

Callback function that indicates the verification/identification template extraction result.

Parameter

Parameter	Description
actionResult	**In**: Always 0. If template extraction fails, the SDK calls the onPalmAtrrib callback function to notify the application.
verTemplate	**In**: Verification/identification template.

Remarks

- When an exception occurs in communication, the SDK calls this callback function repeatedly to notify the application about the exception.
- It is recommended to call the resetEx function when this callback function is triggered for the first time.
- Then, the SDK listens to the USB unplug/plug notification and reconnects the device.
- Click here to view the Function List.

onEnroll

Function Syntax

```
public void onEnroll
        (
                int actionResult,
                int times,
                byte[] verTemplate,
                byte[] regTemplate
                        )
```

Description

Callback function that indicates the palm registration status or result.

Parameter

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>actionResult</td><td>In: Registration status:
<table><tr><td>0</td><td>Registering</td></tr><tr><td>1</td><td>Registered successfully</td></tr><tr><td>Other values</td><td>Failed to be registered</td></tr></table></td></tr>
<tr><td>times</td><td>In: When actionResult = 0/1, this parameter indicates the number of successes in collecting pre-registered templates.

In other cases, this parameter can be ignored</td></tr>
<tr><td>verTemplate</td><td>In: Verification/identification template.
<table><tr><td>actionResult = 0</td><td>The specified verification/identification template can be used to determine whether the current palm has been registered through 1:N identification.</td></tr></table></td></tr>
<tr><td>regTemplate</td><td>In:
<table><tr><td>actionResult = 1</td><td>This parameter indicates a registered template.
In other cases, the parameter value is null.</td></tr></table></td></tr>
</table>

Remarks

- After startEnroll is called, the SDK calls this callback function to notify the application about the palm registration result.
- Click here to view the Function List.

onFeatureInfo

Function Syntax

```
public void onFeatureInfo
        (
                int actionResult,
                int imageQuality,
                int tempalteQuality,
                int[] roi
```

)

Description

Callback function that indicates the palm feature status.

Parameter

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>actionResult</td><td>In:
<table><tr><td>0</td><td>Indicates that palm features have been extracted successfully</td></tr><tr><td>Other values</td><td>Indicate a failure to extract palm features, and in this case the application will prompt the user to adjust the position or gesture of the palm.</td></tr></table></td></tr>
<tr><td>imageQuality</td><td>In: When actionResult = 0, this parameter indicates the image quality.
In other cases, this parameter can be ignored</td></tr>
<tr><td>tempalteQuality</td><td>In: When actionResult = 0, this parameter indicates the template quality.
In other cases, this parameter can be ignored</td></tr>
<tr><td>roi</td><td>In: When actionResult = 0, this parameter indicates the palm coordinates.
In other cases, the parameter value is null</td></tr>
</table>

Remarks

- After capturing palm images and extracting the template, the SDK calls this callback function to notify the application about the template extraction result and palm features.
- Click here to view the Function List.

4 Library Description

You can use the following device or algorithm interfaces if you want to use the device or algorithm more flexibly or only need to call either the device SDK or algorithm SDK of RMATURA. For details about the palm detection process and registration process, see sections 4.6.1 and 4.6.2.

4.1 Device Library

4.1.1 PalmFactory.class

It is a factory class used to instantiate or destroy a PalmSensor object.

Function List

Class/Interface	Description
com.armatura.android.biometric.device.palmsensor.PalmFactory.class	Palmsensor factory class
createPalmSensor	Instantiates a PalmSensor object
destroy	Destroys a PalmSensor object

createPalmSensor

Function Syntax

```
public static PalmSensor createPalmSensor
        (
                Context context,
                TransportType transportType,
                Map<String, Object> parameters
                        );
```

Description

Creates a PalmSensor instance.

Parameter

Parameter	Description
context	**In**: Application context (getApplicationContext)
transportType	**In**: Type of the transfer protocol (TransportType.USB)
parameters	**In:** USB vendor ID and product ID

Returns

PalmSensor object

Example

```
...
private final static int VID = 0x34c9; //AMT USB vendor ID
private final static int PID = 0x0400;
...
{
...
// Start palm sensor
Map deviceParams = new HashMap();
//set vid
deviceParams.put(ParameterHelper.PARAM_KEY_VID, VID);
//set pid
deviceParams.put(ParameterHelper.PARAM_KEY_PID, PID);
palmsensor = PalmFactory.createPalmSensor(getApplicationContext(),
TransportType.USB, deviceParams);
...
}
```

Remarks

- Click here to view the Function List.

destroy

Function Syntax

public static void destroy(Palmsensor device)

Description

Destroys an object.

Parameter

Parameter	Description
device	**In**: PalmSensor object

Remarks

- Click here to view the Function List.

4.1.2 PalmSensor.class

It is a device interface class that enables operations on the device, such as turning on or off the device, obtaining the image width, device serial number, firmware version and other device attributes, and resetting of device.

Function List

Class/Interface	Description
getSDK_Version	Gets the SDK version
open	Turns on the palm recognition device
close	Turns off the palm recognition device
getImageWidth	Gets the palm image width
getImageHeight	Gets the palm image height
getSerialNumber	Gets the serial number of the device
getFirmwareVersion	Gets the firmware version running on the device
setPalmCaptureListener	Sets the image capturing callback listener interface
startCapture	Starts image capturing
stopCapture	Stops image capturing
reset/resetEx	Resets the device
setParameter	Sets parameters

getSDK_Version

Function Syntax

public String getSDK_Version()

Description

Gets the SDK version.

Returns

SDK version

Remarks

- Click here to view the Function List.

open

Function Syntax

public void open(int index) throws PalmException

Description

Turns on the device.

Parameter

Parameter	Description
index	**In**: Device index (enter 0)

Remarks

- An exception is thrown when the function fails to execute.
- It is recommended to get the USB permission beforehand. For details, see the permission configuration.
- Click here to view the Function List.

close

Function Syntax

public void close(int index) throws PalmException

Description

Turns off the device.

Parameter

Parameter	Description
index	**In**: Enter 0

Remarks

- An exception is thrown when the function fails to execute.
- Click here to view the Function List.

getImageWidth

Function Syntax

public int getImageWidth()

Description

Gets the image width.

Returns

Image width

Remarks

- Click here to view the Function List.

getImageHeight

Function Syntax

public int getImageHeight()

Description

Gets the image height.

Returns

Image height

Remarks

- Click here to view the Function List.

getSerialNumber

Function Syntax

public String getSerialNumber()

Description

Gets the serial number of the device.

Returns

Serial number of the device.

Remarks

- Click here to view the Function List.

getFirmwareVersion

Function Syntax

public String getFirmwareVersion()

Description

Gets the firmware version.

Returns

Firmware version.

Remarks

- Click here to view the Function List.

setPalmCaptureListener

Function Syntax

public void setPalmCaptureListener(int index, PalmCaptureListener listener)

Description

Sets the image capturing listener.

Parameter

Parameter	Description
index	**In:** Enter 0
listener	**In**: PalmCaptureListener (See the description of PalmCaptureListener.class.)

Remarks

- Click here to view the Function List.

startCapture

Function Syntax

public void startCapture(int index) throws PalmException

Description

Starts image capturing (PalmCaptureListener.onCapture is called to notify the application about the result of every capturing operation.).

Parameter

Parameter	Description
index	**In:** Enter 0

Remarks

- Click here to view the Function List.

stopCapture

Function Syntax

public void stopCapture(int index) throws PalmException

Description

Stops image capturing.

Parameter

Parameter	Description
index	**In:** Enter 0

Remarks

- Click here to view the Function List.

reset

Function Syntax

public void reset(int index) throws PalmException

Description

Resets the device.

Parameter

Parameter	Description
index	**In:** Enter 0

Remarks

- Click here to view the Function List.

resetEx

Function Syntax

public void resetEx(int index) throws PalmException

Description

Resets the device.

Parameter

Parameter	Description
index	**In:** Enter 0

Remarks

- This function is equivalent to functions of open, reset, and close executed in sequence.
- Click here to view the Function List.

controlLED

Function Syntax

public boolean controlLED(int index, int ledIndex, int timeout)

Description

Controls status LEDs.

Parameter

Parameter	Description
index	**In:** Enter 0
ledIndex	**In:** 0 Turns off all LEDs. 1 Turns on the red LED. 2 Turns on the green LED. 3 Turns on the blue LED.
timeout	**In:** Timeout period after which the device will turn off the status LEDs Value range: 0-255 (unit: 100 ms)

Returns

true	Success
false	Failure

Example

```
// Turn on the red LED and keep it on for 500 ms
int ledIndex = 1;// Red LED on
int timeout = 5; // Off 500 ms later
palmsensor.controlLED(0, ledIndex, timeout);
```

Remarks

- This function is equivalent to functions of open, reset, and close executed in sequence.
- Click here to view the Function List.

setParameter

Function Syntax

public int setParameter(int index, int code, byte[] paramValue, int size)

Description

Sets parameters.

Parameter

Parameter	Description
index	**In:** Enter 0
code	**In:** Parameter code
paramValue	**In**: Parameter value
size	**In**: Length of the parameter value

Returns

0	success
Other Values	failure

Remarks

- Click here to view the Function List.

4.1.3 PalmCaptureListener.class

It is an image capturing callback listener class that provides the capturing status and device exceptions.

Function List

Class/Interface	Description
com.armatura.android.biometric.device.palmsensor.PalmCaptureListener.class	Image capturing callback listener interface class
onCapture	Callback function that indicates the image capturing result
onException	Callback function that indicates an exception on the device

onCapture

Function Syntax

void onCapture(int actionResult, byte[] palmImage)

Description

Notifies the application of the image capturing result.

Parameter

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>actionResult[in]</td><td>In: The value 0 indicates that the image has been captured successfully.</td></tr>
<tr><td>palmImage[in]</td><td>In:<table><tr><td>actionResult = 0</td><td>This parameter indicates the gray scale of the palm image in the Gray256 system</td></tr><tr><td>actionResult! = 0</td><td>The parameter value is null (no action required)</td></tr></table></td></tr>
</table>

Remarks

- Click here to view the Function List.

onException

Function Syntax

void onException()

Description

Notifies the application of an exception occurring on the device.

Remarks

- It is recommended to call resetEx in this function, and then monitor USB unplug/plug notifications to reconnect the device.

- Do not call the stopCapture function directly in deviceException.
- stopCapture will wait until the thread with the deviceException ends. Therefore, it is recommended to create a new thread to reconnect the device.
- Click here to view the Function List.

4.2 Algorithm Library

4.2.1 AMTPalmService12.class

It is a palm recognition algorithm interface class.

Function List

Class/Interface	Description
com.armatura.amtinfraredservice.irpalm.AMTPalmService12.class	Palm recognition algorithm interface class
com.armatura.amtinfraredservice.irpalm.AMTPalmExtractResult.class	Feature extraction result object
getSDKVersion	Gets the SDK version
getUKeyVendorID	Gets the vendor ID of the USB dongle
getUKeyProductID	Gets the product ID of the USB dongle
init	Initializes the algorithm
free	Releases algorithm resources
extract	Extracts a template
verify	Performs 1:1 palm verification
mergeRegTemplate	Merges pre-registered templates into the registered templates
dbAdd	Adds a registered template to the 1:N identification database
dbDel	Removes the specified template from the database
dbClear	Deletes all templates from the database
dbCount	Gets the total number of templates stored in the database
dbIdentify	Performs 1:N palm identification
dbVerify	Performs 1:1 palm verification

AMTPalmExtractResult

Function Syntax

```
public class AMTPalmExtractResult
    {
        public int result;//Return value
        public int[] rect;
        public int imageQuality;
        public int templateQuality;
        public byte[] preRegTemplate;
        public byte[] verTemplate;
                }
```

Description

Description of data returned after a Template Extraction Operation.

Parameter

Parameter	Description
rect	Palm coordinates; rect[0],rect[1]: Coordinates of the upper left corner (x1,y1) rect[2],rect[3]: Coordinates of the upper right corner (x2,y2) rect[4],rect[5]: Coordinates of the lower right corner (x3,y3) rect[6],rect[7]: Coordinates of the lower left corner (x4,y4)
imageQuality	Image quality
templateQuality	Template quality
preRegTemplate	Pre-registered template
verTemplate	Verification/identification template

Returns

public int result;

0	Template extracted successfully
-11001	Invalid parameter (rawImage is a null pointer or the extractType is set incorrectly.

-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

getSDKVersion

Function Syntax

public static String getSDKVersion()

Description

Gets the SDK version.

Returns

Version number, such as 12.7.0\t1000

Remarks

Click here to view the Function List.

getUKeyVendorID

Function Syntax

public static int getUKeyVendorID()

Description

Gets the vendor ID of the USB dongle.

Returns

Vendor ID

Remarks

- Click here to view the Function List.

getUKeyProductID

Function Syntax

public static int getUKeyProductID()

Description

Gets the product ID of the USB dongle.

Returns

Product ID

Remarks

- Click here to view the Function List.

init

Function Syntax

public static int init(Context context, int width, int height)

Description

Initializes the algorithm.

Parameter

Parameter	Description
context	**In:** Application context
width	**In:** Image width
height	**Out:** Image height

Returns

0	Success
-11001	Parameter error
-11006	Dongle connection failure
Other values	Initialization failure (for reasons such as memory allocation failure)

Remarks

- The image size can be 480 x 640 or 720 x 1280.
- It is recommended to get the USB permission for the dongle before initializing the algorithm.
- Click here to view the Function List.

free

Function Syntax

public static void free()

Description

Releases algorithm resources.

Remarks

- Click here to view the Function List.

extract

Function Syntax

public static AMTPalmExtractResult extract(int extractType, byte[] rawImage)

Description

Extracts a template from Gray256 data.

Parameter

Parameter	Description	
extractType	**In:**	
	1	Extracts pre-registered and verification/identification templates to complete the registration process.
	2	Extracts verification/identification templates to complete

	the verification/identification process. In this case, the value of AMTPalmExtractResult.preRegTemplate is null
rawImage	**In:** Gray256 data

Returns

AMTPalmExtractResult object.

Remarks

- A higher quality of palm images is required for extracting registered templates.
- Click here to view the Function List.

mergeRegTemplate

Function Syntax

```
public static int mergeRegTemplate
        (
                byte[] preRegTemplates,
                int count,
                byte[] regTemplate
                        )
```

Description

Merges five pre-registered templates into the registered template.

Parameter

Parameter	Description
preRegTemplates	**In:** Bytes of the five merged pre-registered templates [5*AMTPalmService12.FIX_PREREG_TEMPLATE_LEN]
count	**In:** Enter 5
regTemplate	**Out**: Registered template, for which the caller needs to apply for memory. The data length is fixed as AMTPalmService12.FIX_REG_TEMPLATE_LEN bytes.

Returns

0	Success

-5	Template merging failure
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Example

```
byte[] preRegTemplates = new
byte[5*AMTPalmService12.FIX_PREREG_TEMPLATE_LEN];
... // Merge five pre-registered templates into preRegTemplates
byte[] regTemplate = new byte[AMTPalmService12.FIX_REG_TEMPLATE_LEN]; //
Memory allocation by the caller
int retVal = AMTPalmService12.mergeRegTemplate(preRegTemplates, 5,
regTemplate);
if （0 == retVal)
{
// Template merging succeeded.
}
```

Remarks

- Click here to view the Function List.

verify

Function Syntax

```
public static int verify(byte[] regTemplate, byte[] verTemplate)
```

Description

Performs palm verification.

Parameter

Parameter	Description
regTemplate	**In:** A registered template (See mergeRegTemplate.)
verTemplate	**In:** A verification template (See AMTPalmExtractResult.verTemplate and other extract functions.)

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- The score range is 0-1000.
- The recommended threshold value is 576.
- Click here to view the Function List.

dbAdd

Function Syntax

public static int dbAdd(String id, byte[] regTemplate)

Description

Adds a registered palm template to the database.

Parameter

Parameter	Description
id	**In:** Palm ID
regTemplate	**In**: A registered template (See mergeRegTemplate.)

Returns

0	Success
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

dbDel

Function Syntax

public static int dbDel(String id)

Description

Removes the palm template of the specified ID from the database.

Parameter

Parameter	Description
id	**In:** Palm ID

Returns

0	Success
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

dbCount

Function Syntax

public static int dbCount()

Description

Gets the total number of palm templates stored in the database.

Parameter

Parameter	Description

id	**In:** Palm ID

Returns

>=0	Total number of palm templates
-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

dbIdentify

Function Syntax

public static int dbIdentify(byte[] verTemplte, String[] id)

Description

Recognizes a palm.

Parameter

Parameter	Description
verTemplte	**In:** A identification template (See extract functions.)
id	**Out:** ID of the palm recognized, for which the caller needs to allocate memory (exp:id = new String[1]).

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- The score range is 0-1000.
- The recommended threshold value is 576.
- Click here to view the Function List.

dbClear

Function Syntax

public static int dbClear()

Description

Clears the database.

Returns

>=0	Score
-11002	Algorithm not initialized

Remarks

- Click here to view the Function List.

dbVerify

Function Syntax

public static int dbVerify(byte[] verTemplate, String id)

Description

Performs palm verification.

Parameter

Parameter	Description
verTemplate	**In:** A verification template (See extract functions.)
id	**In:** Palm ID.

Returns

>=0	Score
-11001	Parameter error (null pointer or insufficient memory)
-11002	Algorithm not initialized

Remarks

- The score range is 0-1000.
- The recommended threshold value is 576.
- Click here to view the Function List.

Appendix

Appendix 1: Glossary

The following definitions will help to understand basic functions of the palm recognition application.

Pre-registered template

Pre-registered templates are only used to merge captured palm features into a registered template. For details, see AMTPalmExtractResult.preRegTemplate.

Verification/Identification template

Verification/Identification templates are palm templates used for 1:1 or 1:N palm recognition. For details, see AMTPalmExtractResult.verRegTemplate.

1:1 Palm Verification

1:1 palm verification, also called palm verification, is a process of verifying whether a user has a valid identity based on the user ID and palm template or determining whether a registered template and several verification templates are extracted from the same palm.

That is, 1:1 biometric verification process authenticates a person's identity by comparing the captured biometric template with a biometric template of that person pre-stored in the database.

1:N Palm Identification

1:N palm identification, also called palm identification, is a process of determining whether a user exists in the system based on the palm of the user, without the user ID. Specifically, the application looks up the palm template database based on the input palm template and returns the name of the user by meeting the threshold, palm similarity degree, and other related information.

So thus, A one-to-many (1:N) biometric identification process instantly compares the person's captured biometric template against ALL stored biometric templates in the system.

Registered template

A registered template is a palm template returned by the mergeRegTemplate interface.

Registered palm

The palm device captures five palm images of the same user to extract pre-registered templates, merges the pre-registered templates into a registered template, and then loads it to the backend database as a registered palm for subsequent palm recognition.

190 Bluegrass Valley Pkwy,

Alpharetta, GA 30005, USA

E-mail: info@armatura.us

www.armatura.us