



API Development Manual:

AMTPalmMobile SDK for Android

API Version: 1.0

Doc Version: 1.0

September 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.

Copyright © 2022 ARMATURA LLC. All rights reserved.

Without the prior written consent of ARMATURA LLC, no portion of this manual can be copied or forwarded in any way or form. All parts of this manual belong to ARMATURA and its subsidiaries (hereinafter the "Company" or "ARMATURA").

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 **AMTPalmMobile 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	6
1.1	OVERVIEW.....	6
1.2	ALGORITHM FEATURES.....	6
1.3	ADVANTAGE OF THE ALGORITHM.....	7
2	TECHNICAL SPECIFICATIONS	8
2.1	ARCHITECTURE.....	9
2.1.1	SDK FILE.....	9
2.1.2	DEVELOPMENT SETUP.....	9
2.2	PROGRAMMING GUIDE.....	9
2.2.1	REGISTRATION PROCESS.....	9
2.2.2	VERIFICATION/IDENTIFICATION PROCESS.....	11
3	SDK INTERFACE DESCRIPTION	14
3.1	TEMPLATE FORMAT.....	14
3.2	INTERFACE DESCRIPTION	14
3.2.1	AMTPALMMOBILE.ARR.....	14
	APPENDIX	28
	APPENDIX 1: ERROR CODE.....	28
	APPENDIX 2: GLOSSARY.....	28
	APPENDIX 3: IMAGE BACKUP DURING REGISTRATION PROCESS.....	29

1 Introduction

This document will provide basic SDK development guide and technical background to help application developers/integrators better understand AMTPalmMobile SDK in their development practice.

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

1.1 Overview

AMTPalmMobile biometric recognition algorithm is AI computer vision-based palm recognition algorithm on true-color RGB images. It not only recognizes and supports palm liveness detect, but also has strong adaptability to various environments of varying lighting condition. It can perform the palm recognition with accuracy even with partially captured or blurred palm images, and less impacted by ambient light. It is fully open to software developers and system integrators, and the SDK can be customized to meet the customer requirements. We keep a consistent model for palm detection, feature extraction, and matching to ensure the compatibility throughout all different SDK versions and cross various platforms.

1.2 Algorithm Features

- **1:N High-Speed Matching Algorithm**
While maintaining the high stability in performance, the algorithm uses a multi-level comparison mode and optimized classifier parameters, to achieve high-speed matching for large-volume users.
- **Palm Quality Assessment**
Evaluate the image quality of the target palm.
- **Highly Secure Anti-Spoofing Protection**
Liveness detection under visible light to ensure the palm is a real and right one, protect the target application from forgery attacks.
- **High-Tolerance to Palm Postures**
The algorithm is not only adaptable to wide Pitch ($\pm 30^\circ$), Yaw ($\pm 45^\circ$), or Roll ($\pm 30^\circ$) angles of palm postures, but also effectively identifies various palm shapes from tensed to bended.

The high posture tolerance allows user to perform palm recognition in a natural and comfortable way, which greatly improves the user experience.

1.3 Advantage of the Algorithm

- The algorithm works well with true color RGB images captured by most common digital cameras for mobile devices or web browsers.
- Simple, intuitive, and developer-friendly programming interfaces.
- Well-documented development guide on code tutorial.
- Rich programming interfaces provide value-added features on applications.

2 Technical Specifications

Development Language

This SDK provides a jar package to support Java development.

Platform Requirements

This SDK supports Android 5.0 or higher.

Technical Parameters

Parameter	Description
Template size	544B
Posture adaptability	Yaw $\leq 45^\circ$, Pitch $\leq 30^\circ$, Roll $\leq 90^\circ$, Bend $\leq 20^\circ$
Palm detection	< 15 ms
Palm feature extraction	< 45 ms
Palm verification/identification (1:10,000)	< 10 ms
Number of palm templates supported	5000
Accuracy	TAR=98.2% when FAR=0.05%

The above performance is based on the tests conducted with the following specifications:

Image resolution: 640x640, CPU: Intel(R) Core(TM) i5-9400 CPU @ 2.90Ghz, RAM: 16GB.

2.1 Architecture

2.1.1 SDK File

- Copy the following file to your Android terminal.

File Name	Description
AMTPalmMobile.arr	Biometric Interface Library

2.1.2 Development Setup

SDK Dynamic-link library files can be copy-paste and installed directly

Please make sure your operating system and computer configuration meet the requirements of software operation before installing the AMTPalmMobile SDK.

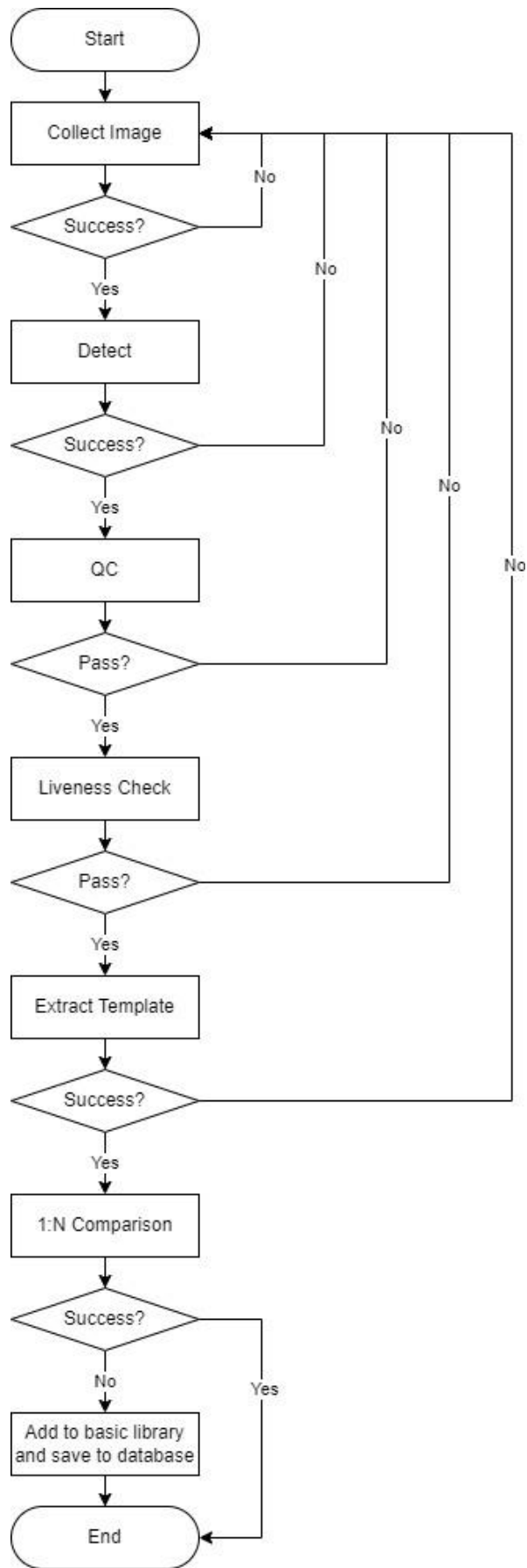
Copy AMTPalmMobile.arr and related files in AMTPalmMobile SDK to the path specified by the user.

2.2 Programming Guide

The following sections will provide introduction and walk-through of the key operating processes and the Biometric registration/comparison processes of the algorithms in AMTPalmMobile SDK for the purpose of further understanding and development.

2.2.1 Registration Process

The extracted palm information can be directly used as the registration template during palm registration. Refer to Section 3 SDK Interface Specification for more information.

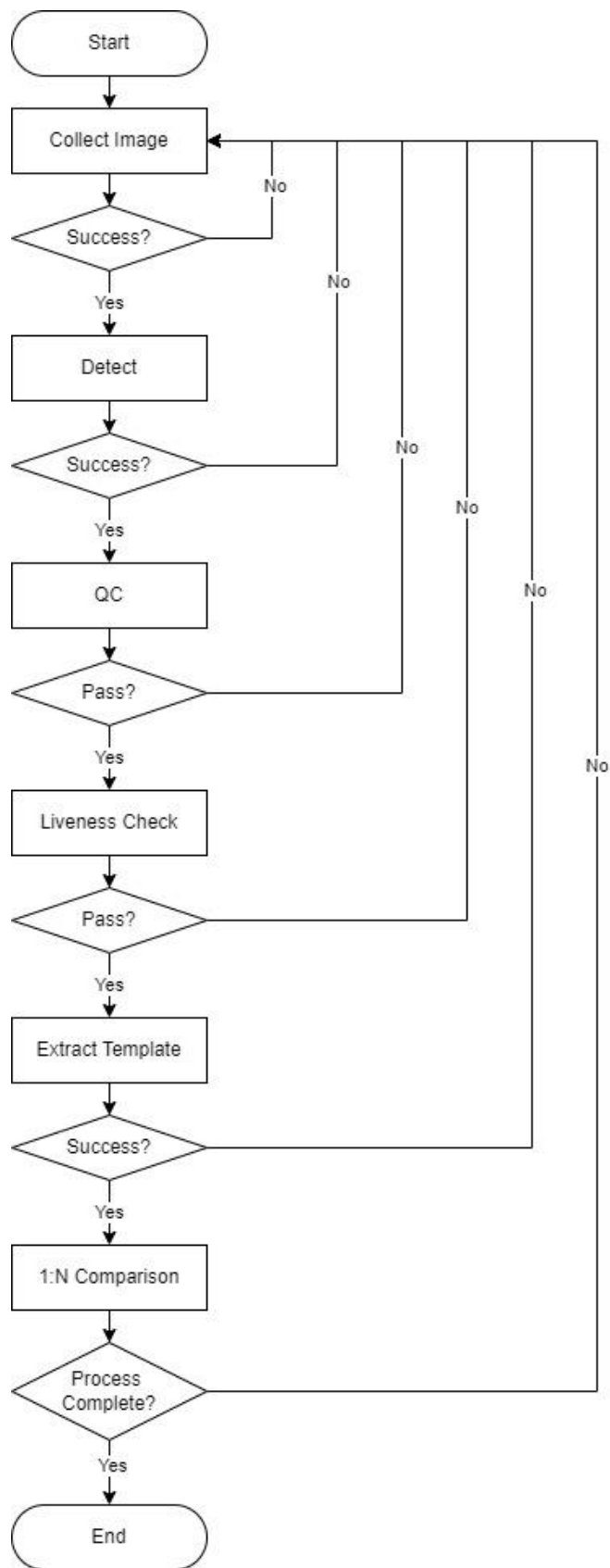


Process Description:

- Call the enrollment class to collect images
- Once the image is successful collected, call Detect to detect the palm
- After the palm image is successfully detected, call GetPalmqIt for quality inspection.
- After passing quality inspection, call GetPalmvL for liveness detection.
- If pass liveness detection (over the liveness threshold value), call GetTemplate to extract the palm template
- After the template is successfully extracted, call DBIdentify to perform 1:N matching to check whether the current template has been registered or not. If it has been registered before, it will prompt the user that the palm has been registered and stops the registration process.
- If the 1:N matching returns with negative value, means no template matched from the database, call DBset to add the palm template to the base library (cache) and save the palm template to the database
- Complete the registration process

2.2.2 Verification/Identification Process

For palm identification (1:N matching), all registered templates need to be added to base library (cache) first. It is recommended to call DBset and add all palm template to the base library after the algorithm is successfully initialized. This process is also recommended for palm verification (1:1 matching).



Process Description:

- Call the enrollment class to collect images
- Once the image is successful collected, call Detect to detect the palm
- After the palm is detected, call GetPalmqlt for quality check.
- After passing quality check, call GetPalmvl for liveness detection.
- If liveness detection returns with positive value, call GetTemplate to extract the template
- Call DBidentify to perform 1:N matching to complete the process.

3 SDK Interface Description

3.1 Template Format

Template Type	Data Length	Description
Palm template	544 Bytes	Work as registration template or verification/identification template

3.2 Interface Description

This is a dynamic library for biometric interface. It is mainly used for palm detection, template extraction, registration, comparison, and palm specification.

3.2.1 AMTPalmMobile.arr

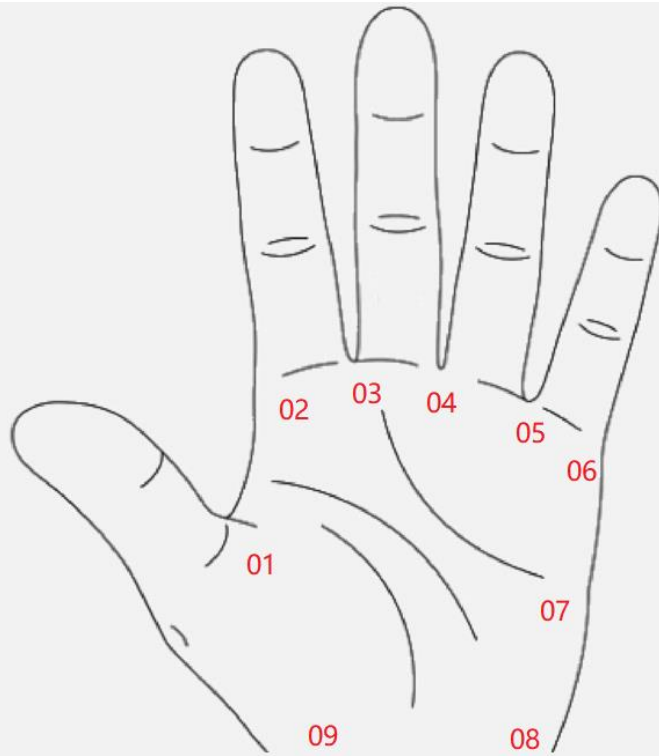
3.2.1.1 Function List

Interface	Description
Version	Get AMPalmMobile SDK version
init	Initialize algorithm resources
Final	Clear algorithm cache
LoadModels	Load models from disk into memory
Detect	Palm Detection
DetectRotation	Palm rotation Detection
GetObject	Get the palm information struct
GetFeature	Extracts palm features
GetTemplate	Extracts palm template
Verify	Perform 1:1 matching

GetPalmvl	Palm liveness check under visible light
DBopen	Connect database
DBclose	Close database
DBset	Store the original template data in the database
DBdelete	Delete specific template from the template database
DBget	Get specific template from the template database
DBcountbyid	Calculate the total number of original palm templates for the specific ID(s)
DBcountid	Calculate the total number of ID assigned in database
DBidentify	Identify in database
DBreset	Clears all data in database
DBverify	Performs a 1:1 matching between specified templates
GetPalmqlt	Get palm quality

3.2.1.2 Description of the structure

```
// Struct for target image information
public class TObject {
    public int class_id;           //a number indicating what is it (Identify target image)
    public int x, y, w, h;         //the bound box of the object
    public float[] point;         //the landmarks of an object
    public float yaw, roll, pitch; //the 3D posture of the object
    public float score;           //the accuracy score of the object
}
// Palm key point coordinates
/*****
/* Landmark[9]
*
```

* Note: The feature points will be marked clockwise in consecutive order for both left and right hands, ignore the position of thumb.

*****/

3.2.1.3 Version

Function Syntax

int Version()

Description

Get the SDK version number.

Parameters

None

Returns

SDK version number

3.2.1.4 init

Function Syntax

```
int init(String db_name)
```

Description

Initializing algorithmic resources.

Parameters

Parameter	Description
db_name[in]	Name and path of database file

Returns

Success when return 0, Failure when return -1

Remarks

1. The above interface should be successfully called before calling any other interface.
2. The algorithm resource will only need to be initialized once during the entire program cycle.

3.2.1.5 Final

Function Syntax

```
int Final()
```

Description

Free up algorithm resources.

Parameters

None

Returns

Success when return value equals zero, error when return value smaller than zero

Remarks

1. The above interface should be called to release algorithm resource when terminating the program.

3.2.1.6 LoadModels

Function Syntax

```
int LoadModels()
```

Description

Loading the model from disk into memory, a time-consuming operation.

Parameters

None

Returns

Success when return value equals zero, error when return value smaller than zero. See error code for more detail (Appendix 1: Error Code)

Remarks

1. Call the above code after initializing with init. Both of them should be successfully called before calling any other interface.

3.2.1.7 Detect

Function Syntax

```
int Detect(byte[] image, int width, int height, String format)
```

Description

Detect target from images, The return value will be the number of detected targets. (Target here means palm).

Parameters

Parameter	Description
image[in]	Image binary data in the specified format
width[in]	Image width
height[in]	Image height
image_format[in]	image format, support: "rgb888", "bgr888", "rgba8888",

	"bgra8888", "bgr565", "nv21", "gray"
--	--------------------------------------

Returns

Success when the returned value is larger than or equal to zero (the number of target detected), error if less than zero. See error code for more detail (Appendix 1: Error Code)

3.2.1.8 DetectRotation

Function Syntax

```
int DetectRotation(byte[] image, int width, int height, String format, int flipx, int flipy, int angle)
```

Description

Detect target from images, The return value will be the number of targets detected. Support image rotation to fix the intended orientation of input image. (target here means palm)

Parameters

Parameter	Description
image[in]	Image binary data in the specified format
width[in]	Image width
height[in]	Image height
image_format[in]	image format, support: "rgb888", "bgr888", "rgba8888", "bgra8888", "bgr565", "nv21", "gray"
angle[in]	rotation angle, supports: 0, 90, 180, 270
flipx[in]	whether the x-axis is flipped, 0 for no flipping, otherwise flipped
flipy[in]	whether the y-axis is flipped, 0 for no flipping, otherwise flipped

Returns

Success when the returned value is larger than or equal to zero (the number of target detected), error if smaller than zero. See error code for more detail (Appendix 1: Error Code).

3.2.1.9 GetObject

Function Syntax

```
TObject GetObject(int index)
```

Description

Get the target information structure

Parameters

Parameter	Description
index[in]	Target index, less than the number of detected targets (0 ~ number of detected targets-1)

Returns

Success when return the target structure, failure when return null.

3.2.1.10 GetFeature

Function Syntax

```
int GetFeature(int index,String name,float[] values,int count)
```

Description

Extracts the features of the target (Target here means palm).

Parameters

Parameter	Description
index[in]	Target index, less than the number of detected targets (0 ~ number of detected targets-1)
name[in]	Type of feature
values[out]	Value of feature
count[out]	Length of feature

Returns

Success when return value equals or larger than zero, error when return value smaller than zero. See error code for more detail (Appendix 1: Error Code)

3.2.1.11 GetTemplate

Function Syntax

```
int GetTemplate(int index,byte[] template)
```

Description

Extract feature template

Parameters

Parameter	Description
index[in]	Target index, less than the number of detected targets (0 ~ number of detected targets-1)
template[out]	Return feature template pointer

Returns

Success when return value equals or larger than zero (the length of template), error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.12 Verify

Function Syntax

```
int Verify(byte[] template1,byte[] template2,float[] score)
```

Description

Compares two templates and returns a similarity score between 0 and 99.3799.

Parameters

Parameter	Description
template1[in]	Input template1
Template2[in]	Input template2
score[out]	Return the similarity scores for pairs of templates

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.13 GetPalmvl

Function Syntax

```
int GetPalmvl(int index,float[] values)
```

Description

Palm liveness check under visible light.

Parameters

Parameter	Description
index[in]	Palm Index
values[out]	The score of liveness detection

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.14 DBopen

Function Syntax

```
int BioFeattrueDBopen(String dbname)
```

Description

Access database.

Parameters

Parameter	Description
dbname[in]	Name of database

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.15 DBclose

Function Syntax

```
int DBclose ( )
```

Description

Close the database.

Parameters

None

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.16 DBset

Function Syntax

```
int DBset(String id, byte[][] templates, int count, int type)
```

Description

Store the original template data in the database. Note: When return value is zero the old database will be switched to new database.

Parameters

Parameter	Description
id[in]	Template id
templates[in]	Multi-template samples
count[in]	Number of templates
type[in]	Identify modal (palm)

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.17 DBdelete

Function Syntax

```
int DBdelete(String id)
```

Description

Delete specific template from database based on template ID.

Parameters

Parameter	Description
id[in]	Target id

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.18 DBGet

Function Syntax

```
int DBget(String id,byte[] template,int size,int type)
```

Description

Read the specified palm template from the template database and place in template in proper order. This function will return the number of valid templates

Parameters

Parameter	Description
id[in]	Input id
template[out]	Output Templates
size[in]	Size of each template
type[in]	Identify modal (palm)

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.19 DBcountbyid

Function Syntax

```
int DBcountbyid(String id,int type)
```

Description

Calculates the number of original palm templates for given ID.

Parameters

Parameter	Description
id[in]	Input id

type[in]	Identify modal (palm)
----------	-----------------------

Returns

Success when return value equals or larger than zero (the length of template), error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.20 DBcountid

Function Syntax

```
int DBcountid()
```

Description

Calculate the total number of ID stored in database.

Parameters

None

Returns

Success when return value equals or larger than zero (the total number of ID stored in database), error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.21 DBidentify

Function Syntax

```
int DBidentify(byte[] template,byte[] id,float[] score,float minscore ,float maxscore,int type)
```

Description

Similarity score identification

Parameters

Parameter	Description
template[in]	Template to be identified
id[out]	Output id
score[out]	Output similarity scores

minscore[in]	minimum score
maxscore[in]	maximum score
type[in]	Identify modal (palm)

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.22 DBReset

Function Syntax

```
int DBreset()
```

Description

Clear all data in database.

Parameters

None

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.23 DBVerify

Function Syntax

```
int DBverify(byte[] template,String id,int type,float[] score)
```

Description

Perform 1 on 1 comparison between specified templates, return similarity scores between 0~99.3799.

Parameters

Parameter	Description
template[in]	Template to be compared
id[in]	ID to be compared

score[out]	Similarity score
type[in]	Identify modal (palm)

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

3.2.1.24 GetPalmqlt

Function Syntax

```
int GetPalmqlt(int index,float[] values)
```

Description

Get palm quality.

Parameters

Parameter	Description
index[in]	Palm Index
values[out]	Quality score of palm image

Returns

Success when return value equals or larger than zero, error when return value smaller than zero, see error code for more detail (Appendix 1: Error Code)

Appendix

Appendix 1: Error Code

Error Code	Description
0	Call succeeded
-1000	Certificate error
-1001	Error reading configuration file
-1002	The feature name is wrong, or the feature is not supported
-1003	Model name error or such model is not supported
-1004	Error identifying ROI (region of interest) name, or the ROI is not supported
-1005	The normalization name is wrong, or the normalization is not supported
-1006	Null pointer error
-1007	Target not detected
-1008	Target index exceeded error
-1009	Input greater than space of temp cache location
-1010	Input parameter error
-1011	Configuration parameter keyword error
-1012	Configuration parameter value error
-1013	Feature type error
-1014	Model type error
-1015	Normalization type error
-10001	Invalid template
-10002	Failed to connect to database or database creation failed
-10003	Failed to access database
-10004	Database access error
-10005	Template size error
-10006	ID not found in database

Appendix 2: Glossary

The following definitions will help our users understand the common functions of biometric identification applications when developing the biometric identification applications.

Verification/Identification template

Verification/Identification templates are used to either 1:1 verification or 1:N identification. The palm templates are obtained by calling the GetTemplate interface.

Registration template

Registration templates are used to registration that is added to the basic library (cache). A registration template is the palm templates returned by calling the **GetTemplate** interface.

Palm Registration

The palm collecting device captures a palm image and then extracts palm template, which is transferred to the backend and stored in database as a registered palm for later palm comparison.

Palm Verification (1:1)

1:1 verification is a process of verifying whether a user has a valid identity based on the user ID and palm template or determining whether the registered template and the verification templates extracted matches the same captured palm image.

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.

Palm Identification (1:N)

1:N 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 database of registered palm templates based on the input palm template and returns the name of the user by meeting the threshold of 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.

Appendix 3: Image backup during registration process

It is recommended to store the image used during registration process. The features may need to be re-extracted when the algorithm model is upgraded

190 Bluegrass Valley Pkwy,

Alpharetta, GA 30005, USA

E-mail: info@armatura.us

www.armatura.us



Copyright © 2022 ARMATURA LLC. All Rights Reserved.