API Development Manual:

AMTPalmLite SDK For Windows

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 **AMT FaceLite SDK For Windwos**.

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 palm 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 standard Win32 API interface and supports C, C++, and C# language development.

Platform Requirements

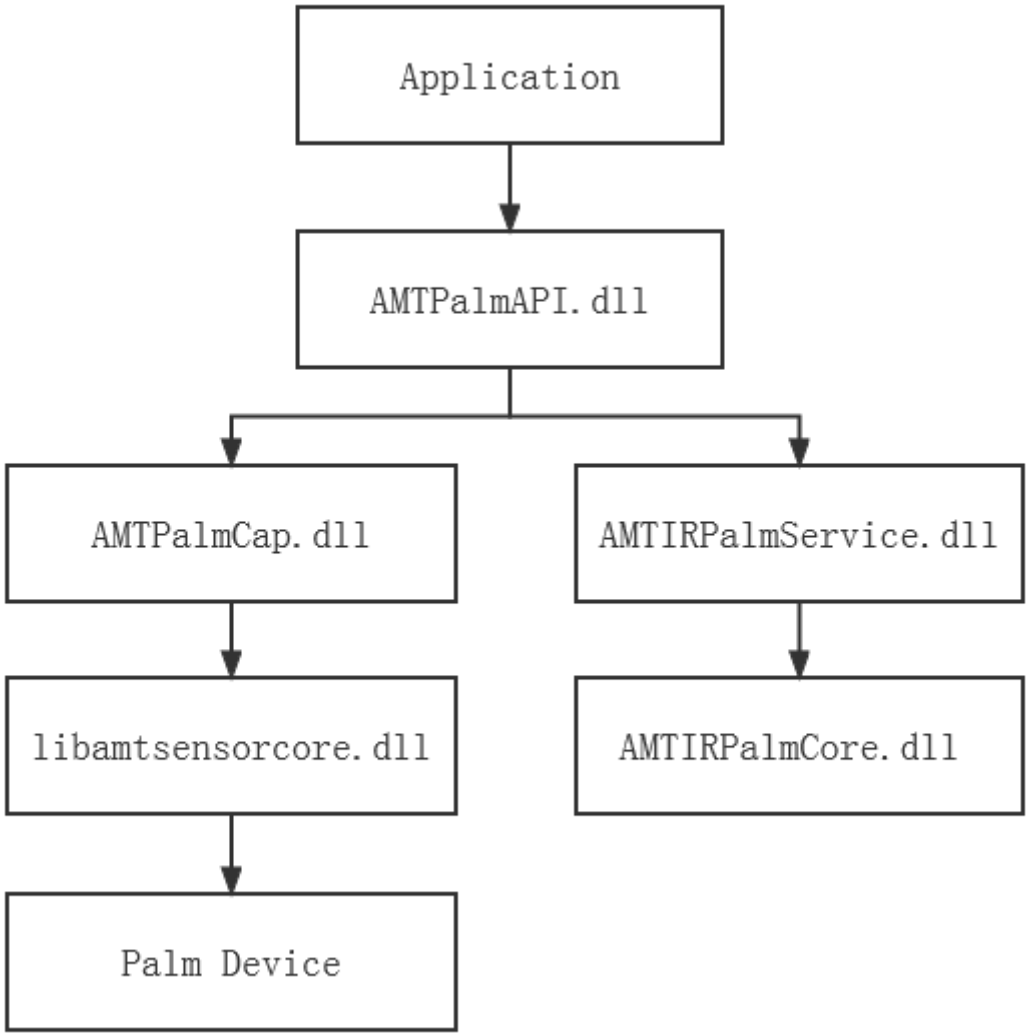
The SDK must be used on 32-bit/64-bit operating system with Windows XP SP3 or higher.

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	< 80 ms
Palm Verification/Identification (1:6000)	< 150 ms
Palm Storage Capacity	6000
Accuracy	TAR = 98.2% when FAR = 0.05%

The preceding algorithm capability indicators are all measured based on the actual image data set (resolution of 480 x 640), 8GB memory and quad-core Inter(R) Core(TM) i5-3210M CPU @2.5GHz processor.

2.1 SDK Architecture



2.1.1 SDK File

Copy the following files (DLL directory) to the Windows terminal.

File Name	Description
libamtsensorcore.dll	Dynamic link library for underlying communication interfaces of the device
AMTIRPalmCore.dll	Palm algorithm core dynamic library
AMTIRPalmService.dll	Dynamic link library for the palm recognition algorithm interface
AMTPalmCap.dll	Dynamic link library for palm capturing
AMTPalmApi.dll	Palm Interface Dynamic Library

2.1.2 Development Setup

SDK dynamic library files can be copied and installed directly

Before installing AMTPalmLite SDK, please make sure that your operating system, computer configuration or Windows mobile terminal device meets the requirements for software operation.

Copy the DLL files such as libamtsensorcore.dll, AMTIRPalmCore.dll, AMTIRPalmService.dll, AMTPalmCap.dll, AMTPalmApi.dll in the AMTPalmLite SDK to the path specified by the user.

2.1.3 USB Information

Palm recognition devices

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

2.2 Programming Guide

The AMTPalmLite SDK provides two sets of APIs to meet different requirements of developers.

AMTPalmApi library is recommended.

- When you use the device library (AMTPalmCap.dll) and algorithm library (AMTIRPalmService.dll), 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 AMTPalmCap and AMTIRPalmService 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

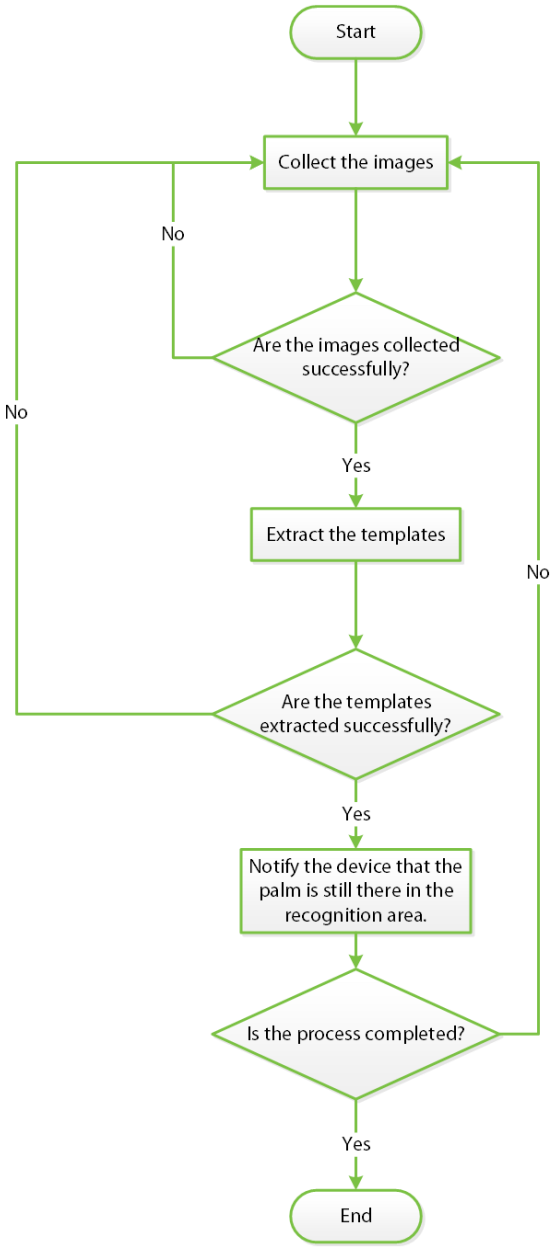
The palm device supports infrared imaging, and the application calls the image capturing function to acquire palm images. When a palm approaches the device, the device turns on the near-infrared LED light, captures palm images, and returns the images. If palm images cannot be captured, the device returns 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 library for SDK integration.

Process Description

- After the palm device is successfully opened, the device API SDK (AMTPalmCap.dll) starts to capture palm images continuously.
- The device API SDK (AMTPalmCap.dll) notifies the application to acquire the image status through the AMT_Palmsensor_Capture function.
- After the AMT_Palmsensor_Capture returns the acquisition, the algorithm API SDK (AMTIRPalmService.dll) is called to extract the template data.
- After extracting the template successfully, it calls the AMT_Palmsensor_SetParameter (hDevice, 2010, 0x03) of the device API (AMTPalmCap.dll) to notify the device that the extraction is successful.

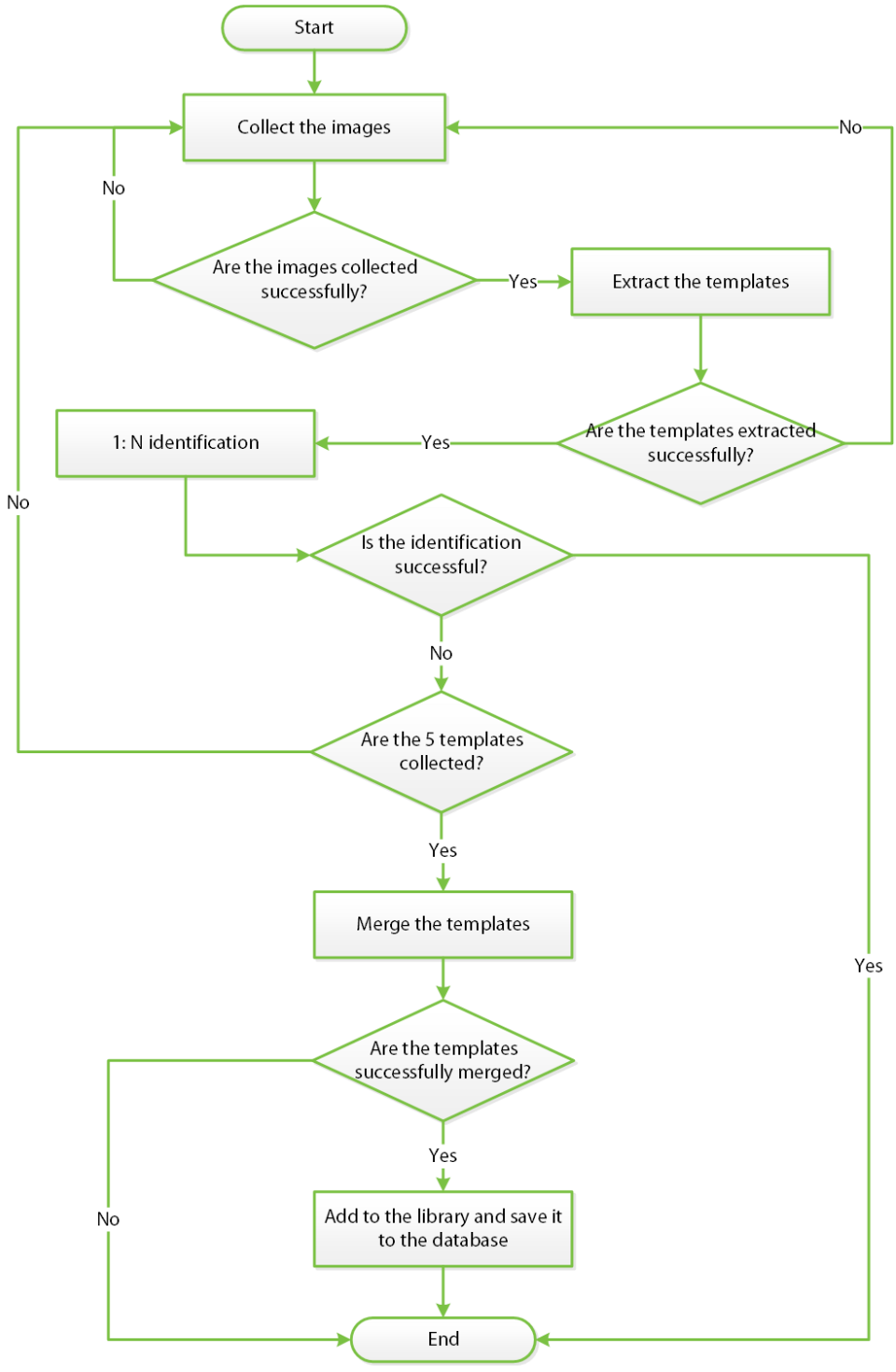
Palm Detection Process Flow



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 Using 1:N Identification



Process Description

- After the palm device is successfully opened, the device API SDK (AMTPalmCap.dll) starts to capture palm images continuously.
- The application layer obtains the acquired image status by calling the device API SDK (AMTPalmCap.dll) AMT_Palmsensor_Capture.
- After the AMT_Palmsensor_Capture returns the acquisition, the AMTPalmServiceExtractTemplate interface of the algorithm API SDK

(AMTIRPalmService.dll) is called to extract the preregistered template and identification template data.

- Call the AMTPalmServiceDBIdentify interface of the algorithm API SDK (AMTIRPalmService.dll) to perform a 1:N identification to determine whether the current template is registered, and if prompted, prompt the user, and end the registration process.
- If less than five templates have been captured, the application continues to capture the next template.
- After collecting five palm templates, the application merges the templates. If the registration fails, the application returns a message and ends the registration process.
- After the registration is successful, call AMTPalmServiceDBAdd of the algorithm API SDK (AMTIRPalmService.dll) to add the registration template to the library and save the registration template to the database.
- The registration process is completed.

2.2.3 Verification/Identification Process

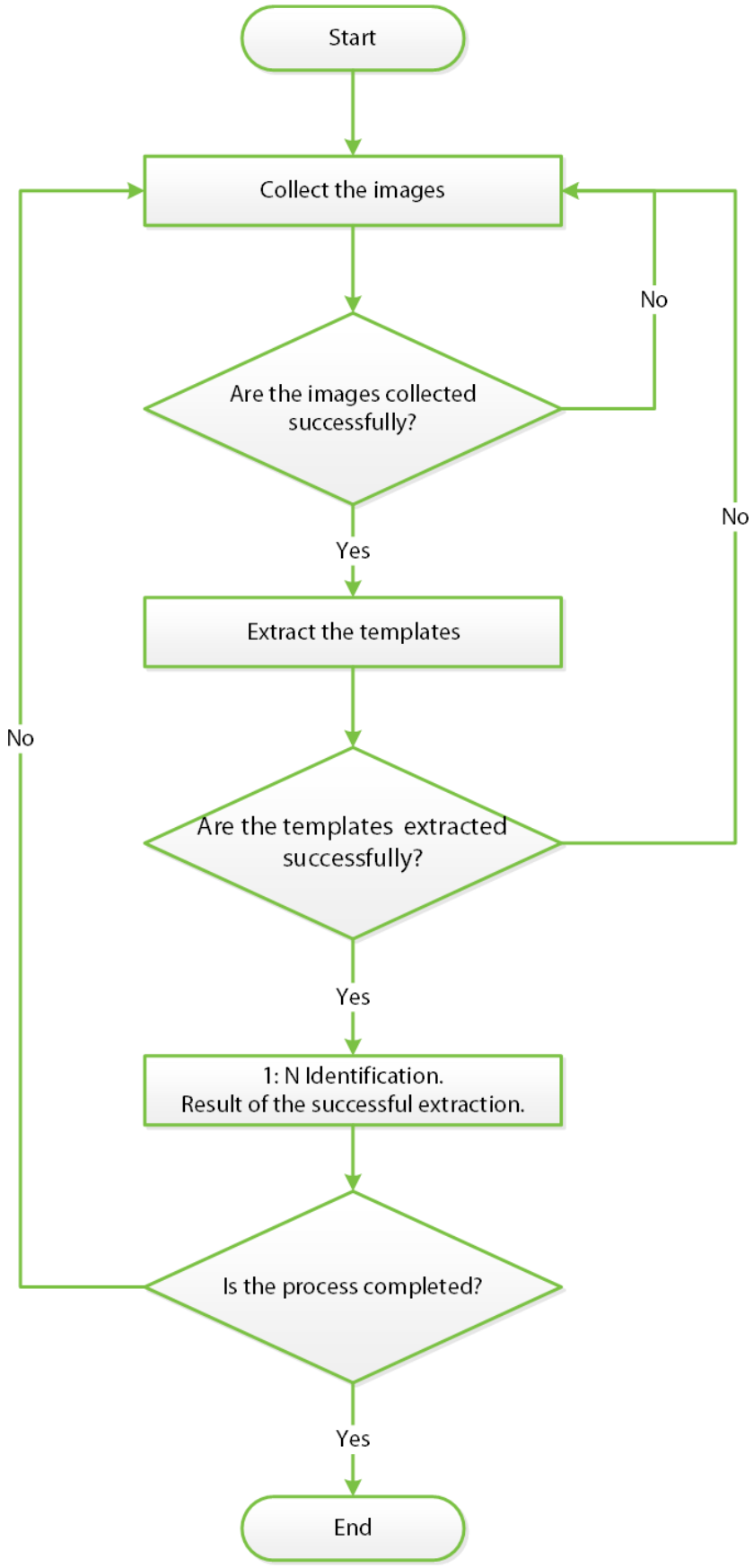
1:N Identification Process

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

Process Description

- After the palm device is successfully opened, the device API SDK (AMTPalmCap.dll) starts to capture palm images continuously.
- The application layer obtains the acquired image status by calling the device API SDK (AMTPalmCap.dll) AMT_Palmsensor_Capture.
- After AMT_Palmsensor_Capture returns the acquisition the AMTPalmServiceExtractTemplate interface of the algorithm API SDK (AMTIRPalmService.dll) is called to extract the identified template data.
- Call the AMTPalmServiceDBIdentify interface of the algorithm API SDK (AMTIRPalmService.dll) for 1:N identification template verification/identification process.
- And once the registered template is identified, the application ends the registration process.

Identification Process Flow



3 SDK Interface Description

3.1 Template Format

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

3.2 AMTPalmApi Library

3.2.1 AMTPalmApi.dll

The AMTPalmApi.dll dynamic library simplifies the calling process of the palm collection library AMTPalmCap.dll and the palm algorithm library AMTIRPalmService.dll interface, helping customers integrate SDK more easily and quickly

Function List

Interface	Description
AMTPalm_Init	Initializes the SDK resources
AMTPalm_Terminate	Releases the SDK resources
AMTPalm_GetVersion	Gets the SDK version
AMTPalm_OpenDevice	Turns on the device and initializes the algorithm
AMTPalm_GetDeviceCount	Gets the number of connected devices
AMTPalm_CloseDevice	Turns off the device and releases the algorithm
AMTPalm_SetParameter	Sets parameters
AMTPalm_GetParameter	Gets parameters
AMTPalm_CapturePalmImageAndTemplate	Collects palm images and extract preregistered template data and verification/identification template data
AMTPalm_Verify	Performs 1:1 palm verification

AMTPalm_VerifyByID	Performs 1:1 verification with the specified id
AMTPalm_MergeTemplates	Consolidated registration template
AMTPalm_DBAdd	Adds a registered template to the database
AMTPalm_DBDel	Removes the specified registered template from the database
AMTPalm_DBCount	Gets the number of registered templates in the database
AMTPalm_DBClear	Clears the database
AMTPalm_DBIdentify	Performs 1:N palm identification

AMTPalm_GetVersion

Function Syntax

int __stdcall AMTPalm_GetVersion(char *version,int size)

Description

Gets the SDK version.

Parameters

Parameter	Description
version	**Out:** Returns the version number (recommended to pre-allocate 64 bytes, sufficient to use)
size	**In**: Memory size allocated by version (number of bytes).

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalm_Init

Function Syntax

int __stdcall AMTPalm_Init ()

Description

Initializes the SDK.

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalm_Terminate

Function Syntax

int __stdcall AMTPalm_Terminate()

Description

Releases the SDK resources.

Returns

0	Success
Other Failure	Error Code

Remarks

Call this function at the end of the program.

Click here to view the Function List.

AMTPalm_OpenDevice

Function Syntax

int __stdcall AMTPalm_OpenDevice(int index,void** handle)

Description

Turns on the device and initializes the algorithm.

Parameter

Parameter	Description
index	**In**: Device index 0~(n-1), n=AMTPalm_GetDeviceCount
handle	**Out:** Device operation instance handle

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalm_GetDeviceCount

Function Syntax

int __stdcall AMTPalm_GetDeviceCount(int *devcnt)

Description

Gets the number of connected devices.

Parameter

Parameter	Description
devcnt	**Out**: Returns the number of devices

Returns

0	Success
Other Failure	Error Code

Remarks: Click here to view the Function List.

AMTPalm_CloseDevice

Function Syntax

int __stdcall AMTPalm_CloseDevice(void *handle)

Description

Turns off the device and releases the algorithm.

Parameter

Parameter	Description
handle	**In**: Device operation instance handle

Returns

0	Success
Other Failure	Error Code

Remarks

Call this function at the end of the program
Click here to view the Function List.

AMTPalm_SetParameter

Function Syntax

```
int __stdcall AMTPalm_SetParameter
        (
                void *handle,
                int paramCode,
                unsigned char* paramValue,
                int size
                        )
```

Description

Sets the parameters.

Parameter

Parameter	Description
handle	**In**: Device operation instance handle
paramCode	**In**: Parameter code (see Parameter Code Description)
paramValue	**In**: Parameter value
size	**In**: Parameter value length

Returns

0	Success
Other Failure	Error Code

Remarks

When setting the status of the external light, please refer to Parameter Code Description.

Click here to view the Function List.

AMTPalm_GetParameter

Function Syntax

```
int __stdcall AMTPalm_GetParameter
        (
                void *handle,
                int paramCode,
                unsigned char * paramValue,
                int *size
                        )
```

Description

Gets the parameters.

Parameter

Parameter	Description
handle	**In**: Device operation instance handle
paramCode	**In**: Parameter code (see Parameter Code Description)
paramValue	**In**: Parameter value
size	**In**: Parameter value pre-allocated buffer size
	Out: Actual parameter value occupied size

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalm_CapturePalmImageAndTemplate

Function Syntax

```
int __stdcall AMTPalm_CapturePalmImageAndTemplate
        (
                void *handle,
                unsigned char*imgBuffer,
                int cbImgBuffer,
                int extractType,
                unsigned char *rawTemplate,
                int *cbRawTemplate,
                unsigned char *verTemplate,
                int *cbVerTemplate,
                int *quality,
                int*pAMTPalmRect,
                void *reserved
                        )
```

Description

Collects the palm images and extracts the pre-registered template data and verification/identification template data.

Parameter

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>handle</td><td>In: Device operation instance handle</td></tr>
<tr><td>imgBuffer</td><td>In: Return palm image data (the original image data with a grayscale image bit depth of 8 bits)</td></tr>
<tr><td>cbImgBuffer</td><td>In: imgBuffer allocated memory size (recommended pre-allocation 480*640)</td></tr>
<tr><td>extractType</td><td>In:
<table>
<tr><td>1</td><td>Represents the extraction of pre-registered features, verification/identification features, mainly used in the registration process</td></tr>
<tr><td>2</td><td>Represents the extraction of verification/identification features, used in the verification/identification process</td></tr>
</table></td></tr>
<tr><td>rawTemplate</td><td>Out: Returns the pre-registered template data (It is recommended to pre-allocate 99120 bytes, and not less than 99120 bytes)</td></tr>
</table>

cbRawTemplate	**In**: memory size allocated by rawTemplate
	Out: actual pre-registered template data size
verTemplate	**Out**: Returns the verification/identification template data (recommended to pre-allocate 27120 bytes, and not less than 27120 bytes)
cbVerTemplate	**In**: the memory size allocated by verTemplate
	Out: the actual verification/identification template data size
quality	**Out**: Returns the quality score of the verification/identification template and pre-registered template
pAMTPalmRect	**Out**: Returns the four coordinate points p0.x p0.y p1.x p1.y p2.x p2.y p3.x p3.y in the counterclockwise order of the palm frame (upper right coordinate point, upper left coordinate point, lower left coordinate points, lower right coordinate points). The application layer allocates an int type array with the array length of 8.
reserved	Reserved parameters, just pass NULL

Returns

0	Success
Other Failure	Error Code

Remarks

- It is recommended to allocate 99120 bytes for palm pre-registration template data, and pre-allocate 27120 bytes for verification/identification templates.
- This interface is a non-thread safe interface.

Click here to view the Function List.

AMTPalm_Verify

Function Syntax

```
int __stdcall AMTPalm_Verify
        (
                void *handle,
                unsigned char *regTemplate,
                int cbRegTemplate,
                unsigned char *verTemplate,
                int cbVerTemplate,
                int *score
                        )
```

Description

Performs 1:1 palm verification.

Parameter

Parameter	Description
handle	**In:** Device operation instance handle
regTemplate	**In:**Registration template data (see AMTPalm_MergeTemplates)
cbRegTemplate	**In:** Registration template data length
verTemplate	**In:** Verification template data (see AMTPalm_CapturePalmImageAndTemplate)
cbVerTemplate	**In**: Verification template data length
score	**Out:** Returns the corresponding verification score (more than 576 can be considered successful)

Returns

0	Success
Other Failure	Error Code

Remarks

- Score range: 0 to 1000
- More than 576, the recognition is successful.

Click here to view the Function List.

AMTPalm_VerifyByID

Function Syntax

```
int __stdcall AMTPalm_VerifyByID
        (
                void *handle,
                unsigned char *verTemplate,
                int cbVerTemplate,
                const char *id,
                int *score
                        )
```

Description

Performs 1:1 verification with the specified id

Parameter

Parameter	Description
handle	**In**: Device operation instance handle
verTemplate	**In**: Verification templates (see AMTPalm_CapturePalmImageAndTemplate)
cbVerTemplate	**In:** Verification template data length
id	**In**: Specified ID
score	**In:** Returns the corresponding verification score (more than 576 can be considered successful)

Returns

0	Success
Other Failure	Error Code

Remarks

- Score range: 0 to 1000
- More than 576, the recognition is successful.

Click here to view the Function List.

AMTPalm_MergeTemplates

Function Syntax

```
int __stdcall AMTPalm_MergeTemplates
        (
                void *handle,
                unsigned char **rawTemplates,
                int mergedCount,
                unsigned char* pMergeTemplate,
                int *cbMergeTemplate
                        )
```

Description

Combines five pieces of palm pre-registered template data into one registered palm template.

Parameter

Parameter	Description
handle	**In:** Device operation instance handle
rawTemplates	**In**: Palm pre-registration template data (multiple templates, it is recommended to pass 5 pre-registration templates, only support up to 5)
mergedCount	**In:** Number of pre-registered template data (It is recommended to pass 5 preregistered templates, and only support up to 5)
pMergeTemplate	**Out:** Palm registration template data after successful merge
cbMergeTemplate	**In**: pMergeTemplate memory allocation size (It is recommended to pre-allocate 8848 bytes, which cannot be less than 8848 bytes)
	Out: Returns the actual pMergeTemplate data length

Returns

0	Success
Other Failure	Error Code

Remarks

- For the palm registration template, it is recommended pre-allocating 8848 bytes
- This interface is a non-thread safe interface.

Click here to view the Function List.

AMTPalm_DBAdd

Function Syntax

```
int __stdcall AMTPalm_DBAdd
        (
                void *handle,
                const char *id,
                unsigned char *pRegTemplate,
                int cbRegTemplate
                        )
```

Description

Adds a registered template to the database (SeeAMTPalm_MergeTemplates).

Parameter

Parameter	Description
handle	**In:** Device operation instance handle
id	**In**: Palm id (<= 24-byte string)
pRegTemplate	**In**: Registration template (see AMTPalm_MergeTemplates)
cbRegTemplate	**In**: Registration template length

Returns

0	Success
Other Failure	Error Code

Remarks

- This interface is a non-thread safe interface.
- Palm id supports up to 24 bytes

Click here to view the Function List.

AMTPalm_DBDel

Function Syntax

int __stdcall AMTPalm_DBDel(void *handle,const char *id)

Description

Deletes the registration template with the specified id from the database.

Parameter

Parameter	Description
handle	**In:** Device operation instance handle
id	**In**: Palm id

Returns

0	Success
Other Failure	Error Code

Remarks

- This interface is a non-thread safe interface.

Click here to view the Function List.

AMTPalm_DBCount

Function Syntax

int __stdcall AMTPalm_DBCount(void *handle,int *count)

Description

Gets the number of registered templates from the database.

Parameter

Parameter	Description
handle	**In:** Device operation instance handle
count	**Out**: The total number of templates returned to the library

Returns

0	Success
Other Failure	Error Code

Remarks

- This interface is a non-thread safe interface.

Click here to view the Function List.

AMTPalm_DBClear

Function Syntax

int __stdcall AMTPalm_DBClear(void *handle)

Description

Clears the database.

Parameter

Parameter	Description
handle	**In:** Device operation instance handle

Returns

0	Success
Other Failure	Error Code

Remarks

- This interface is a non-thread safe interface.

Click here to view the Function List.

AMTPalm_DBIdentify

Function Syntax

```
int __stdcall AMTPalm_DBIdentify
        (
                void * handle,
                unsigned char *verTemplate,
                int cbVerTemplate,
                char * id,
                int *score,
                int minScore,
                int maxScore
                        )
```

Description

Performs 1:N palm identification.

Parameter

Parameter	Description
handle	**In:** Device operation instance handle
verTemplate	**In:** Identification templates (see AMTPalm_CapturePalmImageAndTemplate)
cbVerTemplate	**In**: Template data length
id	**Out:** Returns the palm ID of the successful recognition (at least 24 bytes are allocated)
score	**Out**: Return score
minScore	**In:** Minimum matching score (identification passing threshold, recommended to set to 576)
maxScore	**In**: The highest number of matches (just pass 1000 directly)

Returns

0	Success
Other Failure	Error Code

Remarks

- This interface is a non-thread safe interface.

- Score range: 0 to 1000
- minScore is recommended to be set to 576

Click here to view the Function List.

4 Underlying Library Description

If you choose to use the device or algorithm more flexibly, you can use the following device or algorithm interfaces or just call ARMATURA's device SDK or algorithm SDK. 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 AMTPalmCap.dll

AMTPalmCap.dll dynamic library is the base library for palm collection. Mainly used for palm image data collection and some device parameter setting and acquisition.

Function List

Interface	Description
AMT_Palmsensor_GetVersion	Gets the SDK version
AMT_Palmsensor_Init	Initializes the collection library
AMT_Palmsensor_Free	Releases the collection library resources
AMT_Palmsensor_GetCount	Gets the number of devices
AMT_Palmsensor_Open	Opens the device
AMT_Palmsensor_Close	Closes the device
AMT_Palmsensor_Capture	Acquires the image data
AMT_Palmsensor_SetParameter	Sets the parameters
AMT_Palmsensor_GetParameter	Gets the parameters
AMT_Palmsensor_SetParameterEx	Setting parameter expansion interface
AMT_Palmsensor_GetParameterEx	Gets the parameter extension interface

AMT_Palmsensor_GetVersion

Function Syntax

int __stdcall AMT_Palmsensor_GetVersion(char *version, int len);

Description

Gets the SDK library version number.

Parameters

Parameter	Description
version	**In:** Buffer pointer (recommended to pre-allocate 64 bytes, enough to use)
len	**In**: buffer length allocated by version.

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMT_Palmsensor_Init

Function Syntax

int __stdcall AMT_Palmsensor_Init();

Description

Initializes the collection library.

Returns

0	Success
Other Failure	Error Code

Remarks

Before calling all functions (except AMT_Palmsensor_GetVersion), please initialize the device list

Click here to view the Function List.

AMT_Palmsensor_Free

Function Syntax

int __stdcall AMT_Palmsensor_Free();

Description

Releases the collection library.

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMT_Palmsensor_GetCount

Function Syntax

int __stdcall AMT_Palmsensor_GetCount();

Description

Gets the number of devices.

Returns

0	No current ~~collector~~device access
>0	Current number of connected ~~collector~~devices

Remarks

Click here to view the Function List.

AMT_Palmsensor_Open

Function Syntax

void* __stdcall AMT_Palmsensor_Open(int index);

Description

Opens the ~~collector~~device.

Parameters

Parameter	Description
index	**In:** Device index 0~(n-1), n=AMT_Palmsensor_GetCount (generally pass 0)

Returns

Device operation instance handle

NULL void*	Open failed
Other void*	Success

Remarks

Click here to view the Function List.

AMT_Palmsensor_Close

Function Syntax

int __stdcall AMT_Palmsensor_Close(void *context);

Description

Closes the ~~collector~~device.

Parameters

Parameter	Description
context	**In:** Device operation instance handle

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMT_Palmsensor_Capture

Function Syntax

```
int __stdcall AMT_Palmsensor_Capture
        (
                void *context,
                unsigned char *imageBuffer,
                int imageBufferSize
                        );
```

Description

Acquires the images.

Parameters

Parameter	Description
context	**In:** Device operation instance handle
imageBuffer	**Out:** Image data collected(>=(width*height) Bytes)
imageBufferSize	**In:** imageBuffer pre-allocated buffer size (generally allocated >= (width*height) bytes is sufficient)

Returns

>0	Success (Returns the actual image data size)
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMT_Palmsensor_SetParameter

Function Syntax

```
int __stdcall AMT_Palmsensor_SetParameter
        (
                void *context,
                int paramCode,
                int paramValue
                        );
```

Description

Sets the parameters.

Parameters

Parameter	Description
context	**In:** Device operation instance handle
paramCode	**In:** Parameter code (see Parameter Code Description)
paramValue	**In:** Parameter value (see Parameter Code Description)

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMT_Palmsensor_GetParameter

Function Syntax

```
int __stdcall AMT_Palmsensor_GetParameter(void *context,int paramCode);
```

Description

Gets the parameters.

Parameters

Parameter	Description
context	**In:** Device operation instance handle
paramCode	**In:** Parameter code (see Parameter Code Description)

Returns

>=0	Success (returns the obtained parameter value)
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMT_Palmsensor_SetParameterEx

Function Syntax

```
int __stdcall AMT_Palmsensor_SetParameterEx
        (
                void *context,
                int paramCode,
                char *paramValue,
                int paramLen
                        );
```

Description

Sets the parameters.

Parameters

Parameter	Description
context	**In:** Device operation instance handle
paramCode	**In:** Parameter code (see Parameter Code Description)
paramValue	**In**: Parameter value (see Parameter Code Description)
paramLen	**In:** Parameter value length

Returns

>=0	int Success

Other Failure	Error Code

Remarks

Click here to view the Function List.

AMT_Palmsensor_GetParameterEx

Function Syntax

```
int __stdcall AMT_Palmsensor_GetParameterEx
        (
                void *context,
                int paramCode,
                char *paramValue,
                int *paramLen
                        );
```

Description

Gets the parameters.

Parameters

Parameter	Description
context	**In:** Device operation instance handle
paramCode	**In:** Parameter code (see Parameter Code Description)
paramValue	**In**: Parameter value (see Parameter Code Description)
paramLen	**In:** Parameter value length
	Out: actual parameter value occupied size

Returns

>=0	int Success
Other Failure	Error Code

Remarks: Click here to view the Function List.

4.2 Algorithm Libraries

4.2.1 AMTIRPalmService.dll

AMTIRPalmService.dll dynamic library is a palm algorithm interface library. Mainly used for palm template extraction, registration, verification/identification, etc.

Function List

Interface	Description
AMTPalmServiceVersion	Gets the palm algorithm version number
AMTPalmServiceInit	Palm algorithm initialization
AMTPalmServiceFinal	Free palm algorithm resources
AMTPalmServiceSetParam	Set parameters
AMTPalmServiceExtractTemplate	Extract pre-registered templates and verification/identification templates from palm images
AMTPalmServiceVerify	Performs 1:1 palm verification
AMTPalmServiceVerifyByID	Performs 1:1 verification with the specified id
AMTPalmServiceMergeTemplates	Consolidated registration template
AMTPalmServiceDBAdd	Adds a registered template to the database
AMTPalmServiceDBDel	Deletes the specified registration template from the database
AMTPalmServiceDBCount	Gets the number of templates in the database
AMTPalmServiceDBClear	Clear the database
AMTPalmServiceDBIdentify	Performs 1:N palm identification

AMTPalmServiceVersion

Function Syntax

```
int __stdcall AMTPalmServiceVersion
        (
                int *versionData,
                int *major,
                int *minor,
                char *version,
                int size
)
```

Description

Gets the SDK library version number.

Parameters

Parameter	Description
versionData	**Out:** Returns the date of the version number
major	**Out**: Major version number.
minor	**Out:** Minor version number
version	**Out**: Return the version number (recommended to pre-allocate 64 bytes, enough to use)
size	**In**: Memory size allocated by the version (number of bytes)

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalmServiceInit

Function Syntax

```
int __stdcall AMTPalmServiceInit
        (
                void** handle,
                int width,
                int height,
                bool IsMallocDb
                        )
```

Description

Palm algorithm initialization.

Parameters

Parameter	Description
handle	**Out:** Algorithm handle
width	**In**: Palm image width (480 can be passed)
height	**In**: Palm image height (only 640 can be passed)
IsMallocDb	**In**: Pass true

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalmServiceFinal

Function Syntax

int __stdcall AMTPalmServiceFinal()

Description

Releases palm algorithm resources.

Returns

0	Success
Other Failure	Error Code

Remarks

Call this function at the end of the program.

Click here to view the Function List.

AMTPalmServiceSetParam

Function Syntax

```
int __stdcall AMTPalmServiceSetParam
        (
                void *handle,
                int paramIndex,
                int paramValue
                        )
```

Description

Sets the parameters.

Parameters

Parameter	Description
handle	**In**: Just pass 0
paramIndex	**In**: Parameter code (see Parameter Code Description)
paramValue	**In:** Parameter value

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalmServiceExtractTemplate

Function Syntax

```
int __stdcall AMTPalmServiceExtractTemplate
        (
                void *handle,
                unsigned char *imgBuffer,
                int extractType,
                unsigned char *rawTemplate,
                int *cbRawTemplate,
                unsigned char *verTemplate,
                int *cbVerTemplate,
                int *quality,
                int *pAMTPalmRect
                        )
```

Description

Extracts the verification/identification templates and registration templates from palm images.

Parameters

Parameter	Description
handle	**In**: Just pass 0
imgBuffer	**In**: Palm image data (original image data with 8-bit grayscale image bit depth)
extractType	**In**: Represents the extraction of pre-registered features, verification/identification features mainly used in the registration process.

	Represents extraction of verification/identification features, used in the verification/identification process
rawTemplate	**Out**: Return pre-registered template data (It is recommended to pre-allocate 99120 bytes, not less than 99120 bytes)
cbRawTemplate	**In**: rawTemplate pre-allocated memory size.
	Out: actual pre-registered template data size.
verTemplate	**Out**: Return verification/identification template data (recommended to pre-allocate 27120 bytes, not less than 27120 bytes)
cbVerTemplate	**In**: VerTemplate pre-allocated memory size
	Out: actual verification/identification template data size
quality	**Out:** Returns the quality score of the verification/identification template and pre-registered template
pAMTPalmRect	**Out:** Return the four coordinate points p0.x p0.y p1.x p1.y p2.x p2.y p3.x p3.y of the palm rectangular frame in the counterclockwise order (upper right coordinate point, upper left coordinate point, lower left Coordinate points, lower right coordinate points). The application layer allocates an int type array with an array length of 8.

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalmServiceVerify

Function Syntax

```
int __stdcall AMTPalmServiceVerify
        (
                void * handle,
                const unsigned char *regTemplate,
                unsigned char *verTemplate
                        )
```

Description

Performs 1:1 palm verification.

Parameters

Parameter	Description
handle	**In**: Just pass 0
regTemplate	**In**: Register template data
verTemplate	**In:** Compare template data

Returns

>0	Returns the corresponding verification score
Other Failure	Error Code

Remarks

- Score range: 0 to 1000
- More than 576, the recognition is successful

Click here to view the Function List.

AMTPalmServiceVerifyByID

Function Syntax

```
int __stdcall AMTPalmServiceVerifyByID
        (
                void *handle,
                const unsigned char *verTemplate,
                const char *id
                        )
```

Description

Performs 1:1 verification with the specified id.

Parameters

Parameter	Description
handle	**In**: Just pass 0
verTemplate	**In:** Verifies the template data
id	**In**: Palm id

Returns

0	Returns the corresponding verification/identification score
Other Failure	Error Code

Remarks

- Score range: 0 to 1000
- More than 576, the recognition is successful

Click here to view the Function List.

AMTPalmServiceMergeTemplates

Function Syntax

```
int __stdcall AMTPalmServiceMergeTemplates
        (
                void *handle,
                unsigned char **rawTemplates,
                int mergedCount,
                unsigned char* pMergeTemplate,
                int *cbMergeTemplate
                        )
```

Description

Combines five pieces of palm pre-registered template data into one registered palm template.

Parameters

Parameter	Description

handle	**In**: Just pass 0
rawTemplates	**In:** Palm pre-registration template data (multiple templates, it is recommended to pass 5 pre-registration templates, only support up to 5)
mergedCount	**In**: Number of pre-registered template data (It is recommended to pass 5 preregistered templates, and only support up to 5)
pMergeTemplate	**Out**: Register template data in the palm after successful merge.
cbMergeTemplate	**In:** pMergeTemplate memory allocation size (It is recommended to pre-allocate 8848 bytes, which cannot be less than 8848 bytes)
	Out: Returns the actual pMergeTemplate data length

Returns

0	Success
Other Failure	Error Code

Remarks

Click here to view the Function List.

AMTPalmServiceDBAdd

Function Syntax

```
int __stdcall AMTPalmServiceDBAdd
        (
                void *handle,
                const char *id,
                unsigned char *regTemplate,
                int count
                        )
```

Description

Adds the registration template to the database.

Parameters

Parameter	Description

handle	**In**: Just pass 0
id	**In**: Palm id
regTemplate	**In**: Register template data
count	**In**: Number of registered template data (pass 1)

Returns

0	Success
Other Failure	Error Code

Remarks

This interface is a non-thread safe interface.

Click here to view the Function List.

AMTPalmServiceDBDel

Function Syntax

int __stdcall AMTPalmServiceDBDel(void *handle, const char *id)

Description

Delete the specified registration template from the 1:N ~~bottom~~base library.

Parameters

Parameter	Description
handle	**In**: Just pass 0
id	**In**: Palm id to delete

Returns

0	Success
Other Failure	Error Code

Remarks

This interface is a non-thread safe interface.

Click here to view the Function List.

AMTPalmServiceDBCount

Function Syntax

int __stdcall AMTPalmServiceDBCount(void * handle)

Description

Gets the total number of templates stored in the database.

Parameters

Parameter	Description
handle	**In**: Just pass 0

Returns

>=0	Returns the number of registration templates stored in the database
Other Failure	Error Code

Remarks

This interface is a non-thread safe interface.

Click here to view the Function List.

AMTPalmServiceDBClear

Function Syntax

int __stdcall AMTPalmServiceDBClear (void *handle)

Description

Clear the database.
Parameters

Parameter	Description
handle	**In**: Just pass 0

Returns

0	Success
Other Failure	Error Code

Remarks

- This interface is a non-thread safe interface.
- Click here to view the Function List.

AMTPalmServiceDBIdentify

Function Syntax

```
int __stdcall AMTPalmServiceDBIdentify
        (
                void *handle,
                const unsigned char *verTemplate,
                char *id,
                int minScore,
                int maxScore
                        )
```

Description

Performs 1:N palm identification.

Parameters

Parameter	Description
handle	**In**: Just pass 0
verTemplate	**In**: identifies the template data
id	**Out**: Returns the matched palm id (at least 24 bytes are allocated)
minScore	**In**: Minimum matching score (identification passing threshold, recommended to set to 576)
maxScore	**In:** Highest match score (just pass 1000 directly)

Returns

>=0	Match score
Other Failure	Error Code

Remarks

- This interface is a non-thread safe interface.
- Score range: 0 to 1000
- minScore is recommended to be set to 576
- Click here to view the Function List.

Appendix

Appendix 1: Glossary

The following definitions will help you understand basic functions of a palm recognition application and complete integrated development of such an application.

Re-registered template

Pre-registered templates are only used to merge captured palm features into a registered template.

Verification/Identification template

Verification/Identification templates are palm templates used for 1:1 or 1:N palm recognition.

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 recognition, 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 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 AMTPalm_MergeTemplates or AMTPalmServiceMergeTemplates interface.

Registered palm

The palm ~~collector~~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.

Appendix 2: AMTPalmApi Library

Parameter Code Description

Parameter code	Data type/length	Attribute	Description
1	Int/4Bytes	R	Gets the palm image width
2	Int/4Bytes	R	Gets the palm image height
1103	Int/4Bytes	R	Gets the serial number of the device
2004	Int/4Bytes	W	Controls the LED status lights
10010	Int/4Bytes	R	Get device firmware version

Remarks

When using parameter code 2004 to control the LED light. The paramValue parameter value needs to be passed in two bytes. The high byte indicates the flash time. The flash time is required to be set with specific value (*100ms). After the set time is exceeded, the device automatically turns off the flash light. The low byte indicates the type of LED light color.

The specific color types of the lights are as follows

Types of LED color code	Description
0	Turns off all LEDs
1	Turns on the red LED
2	Turns on the green LED
3	Turns on the blue LED

Examples

Set to light up red, and the light-up time is: 500ms

Function Syntax

```
int nParamValue = 0;
int nLedTime = 5;//The actual flash time of the device is (5*100ms)
int nRedLed = 1;//Red light
nParamValue = (nLedTime<<8) + nRedLed;
int paramSize = sizeof(int);
int ret = AMTPalm_SetParameter(m_hDevice,2004,(unsigned
char*)&nParamValue,paramSize);
```

Get the width and height of the palm image

Function Syntax

```
int paramSize = sizeof(int);
int nWidth = 0;
int nHeight = 0;
int ret = AMTPalm_GetParameter(m_hDevice, 1, (unsigned char*)&nWidth, &paramSize);
paramSize = sizeof(int);
int ret = AMTPalm_GetParameter(m_hDevice, 2, (unsigned char*)&nHeight,
&paramSize);
```

Error Code

Error code	Description
0	Successful operation
-1	Operation failed
-2	Device is not connected
-3	Null pointer
-4	Invalid parameter
-5	Interface is not supported
-6	Failed to initialize algorithm library
-7	Invalid handle
-8	No palm detected
-9	Insufficient software memory allocation
-13	Failed to extract template
-14	Failed to load dynamic library of palm algorithm
-15	Incorrect template format
-16	Failed to add registration template to the library (algorithm allocates memory error)
-17	Template conversion failed
-18	The verification/identification template data to be synthesized is incorrect
-19	Failed to open device
-103	No such id in the database (1:N ~~bottom~~base library does not have this id)
-105	The characteristics of the id of the database are invalid (1: N ~~bottom~~base library)
-106	Duplicate id added
-200	The database is full (1:N ~~bottom~~base library)
-1000	Dongle error

Appendix 3: Device Library

Parameter Code Description

Parameter code	Data type/length	Attribute	Description
1	Int/4Bytes	R	Gets the palm image width
2	Int/4Bytes	R	Gets the palm image height
1103	Int/4Bytes	R	Gets the serial number of the device
2004	Int/4Bytes	W	Control LED status lights
2010	Int/4Bytes	W	Notifies the device if palm is away from the recognition area
2012	Int/4Bytes	W	Set the time-out period of the near infrared fill light off
10010	Int/4Bytes	R	Get device firmware version

Remarks

- The parameter code 2010 is used to notify the device whether the palm is away from the recognition area, and the paramValue parameter value: 0x03 means there is a palm placed in the recognition area, 0x04 means that the palm is away from the recognition area.
- The parameter code 2012 is used to set the near-infrared LED light off timeout. The timeout unit is: 1000ms (that is, the set value * 1000ms)
- The parameter code 2004 is used to control the LED light. The paramValue parameter value needs to be passed in two bytes. The high byte indicates the flash time. The flash time is the specific value to be set (*100ms). After the set values of the flash time is exceeded, the device automatically turns off the light. The low byte indicates the type of LED light color

The specific color types of the lights are as follows

Types of LED colour code	Description
0	turns off all LEDs
1	turns on the red LED
2	turns on the green LED
3	turns on the blue LED

Examples

Set to light up red, and the light-up time is: 500ms

Function Syntax

```
int nParamValue = 0;
int nRedLed = 1;//Red light
nParamValue = nRedLed;
int ret = AMT_Palmsensor_SetParameter(m_hDevice,2004,nParamValue);
```

Get the width and height of the palm image

Function Syntax

```
int nWidth = AMT_Palmsensor_GetParameter(m_hDevice, 1);
int nHeight = AMT_Palmsensor_GetParameter(m_hDevice, 2);
```

Notify the device if the palm is away from the recognition area)

Function Syntax

```
//Notifies the device that the palm is away from the recognition area
int nParamValue = 0x04;
int ret = AMT_Palmsensor_SetParameter(m_hDevice,2010, nParamValue);
//Notifies the device that there is a palm placed in the recognition area
nParamValue = 0x03
ret = AMT_Palmsensor_SetParameter(m_hDevice, 2010, nParamValue);
```

Get the firmware version number

Function Syntax

```
int ret = AMT_Palmsensor_GetParameter(pHandle->hDevice, 10010);
char szFWVersion[24] = { 0x0 };
// High byte is the major version number
int major = (ret & 0xFF00) >> 8;
// The low byte is the minor version number
int minjor = ret & 0xFF;
int fwVerLen = sprintf(szFWVersion,"V%d.%d",major,minjor);
```

Error Code

Error code	Description
0	Successful operation
-1	No device found
-2	Not supported (wrong parameter code)
-3	Invalid handle
-4	Error pointer (null pointer)
-5	Wrong parameter value
-6	Insufficient software memory allocation
-7	Data check error (wrong data tail)

Appendix 4: Algorithm Library

Parameter Code Description

Parameter code	Data type/length	Attribute	Description
100	Int/4Bytes	W	Sets the template extraction detection threshold

Error Code

Error code	Description
0	Successful operation
-1	Incorrect parameters, no palm detected, low palm quality, and feature extraction failure
-3	Null pointer
-4	Invalid parameter
-6	Failed to initialize algorithm library
-9	Insufficient memory allocated by software
-14	Failed to load dynamic library of palm algorithm library
-103	No such id in the database (1:N base library without this id)
-105	The characteristics of the id of the database are invalid (1: N ~~bottom~~base library)
-106	Duplicate id added
-200	The database is full (1:N base library)
-1000	Dongle error

190 Bluegrass Valley Pkwy,

Alpharetta, GA 30005, USA

E-mail: info@armatura.us

www.armatura.us