suprema
BIOMETRICS & SECURITY

BioMini SDK for Windows

Programming Manual

Content

Chapter 1. INTRODUCTION

The BioMini SDK provides the specific functions to capture fingerprints, to display scanner real-time images on the screen while the finger is on the platen and includes all PC interfaces and drivers.

1.1. What's New

Version 3.9.1

- Compatible with new Suprema fingerprint scanners
  - BioMini Slim 3 (BM-Slim3)
- Template quality performance improvement
- Function improvement
  - BioMini Slim 2 image enhancement
  - timeout error code changed from -211 to -241

1.2. History

Version 3.8.0

- Improvement in checking License validity
- Improvement in stability

Version 3.7.5

- Added a software-based LFD feature for below devices (UFS_SetParameter)
  - BioMini Slim (SFU-S20)
  - BioMini Plus 2 (SFU-550)
  - BioMini Slim 2 (BM-Slim2)

Version 3.7

- Compatible with new Suprema fingerprint scanners
  - BioMini Slim 2 (BM-Slim2)
- UFScanner - New function added
  - NFIQ 2.0(NIST Fingerprint Image Quality) mode (UFS_SetParameter)
- Added a Java sample project

Version 3.6

- Compatible with new Suprema fingerprint scanners
  - BioMini Plus2 (SFR550)
- UFScanner - UFS_CaptureSingleUI function added
  - Performs same as UFS_CaptureSingle and Popup Window appears
- UFMatcher - UFM_VerifyEx function added
  - Performs same as UFM_Verify, and returns matching score by 6th parameter

Version 3.5.5

- Do not check license file(UFLicense.dat file removed)
- UFScanner - New APIs added
  - 3/330 BioMin SDK for Windows v3.8.0
  - Save WSQ Image Buffer to variable size
  - Decompress WSQ format data to data of Bmp format
  - Get quality information of image(UFS_GetFPQuality)
  - Added EnrollUI, VerifyUI APIs
- UFDatabase is removed

Version 3.5.0

- Compatible with new Suprema fingerprint scanners
  - BioMini Slim (SFR600), BioMini Combo and SFU-S20
- UFScanner - UFS_StartAutoCapture added
  - Auto finger detect and capture function
- UFScanner - UFS_ExtractEx function added
  - Template extraction function for large size template
- UFScanner - UFS_SelectTemplateEx added
  - Template selection function for large size template

Version 3.4.2

- The SentinelHASP Basic USB dongle added for license control

Version 3.4.1

- Java JNA module fixed
  - Call the UFM_Identify_J instead of the UFM_Identify when using JNA module

Version 3.4

- Universal manufacturer ID embedded in scanners without license control

Version 3.3.1

- Support new sensor, which has AGC(Auto Gain Control) feature on BioMini
- Removed brightness setting from new sensor since the AGC automatically finds it's optimal brightness setting

Version 3.3.0

- Updated extraction and matching algorithm is applied
  - The performance on the low quality fingerprints is improved
- Applied advanced quality measure in extraction function
- Modified basic samples to use 1 template or 2 templates when enrollment
- Support JAVA JNI wrapper
- Added JAVA JNI demo sample
- Added latent fingerprint defense for BioMini
- Support ISO19794-4 image format saving in BioMini SDK
- Support WSQ image format saving in BioMini SDK

Version 3.2.0

- Support BioMini Plus - High quality sensor scanner
- LFD(live fingerprint detection) algorithm - the live finger detection algorithm.
- Support feature drawing function in BioMini SDK - UFScanner
- Update sample demo of SDK and sample demo usage tutorial of manual

Version 3.1.0

- Support Java interfaces
- Support ANSI378 templates

Version 3.1.0.6

- Added BioMini Lock to Image SDK- License lock mode

Version 3.0.0

- Completely new interface compared with version 2.x
- Support full functionality for managing scanners
  - Support handling multiple devices
  - Support plug-and-play events
- Ensure thread safety for all functions
- Scanner, matcher, extractor and database are treated as independent objects

Version 3.0.0.15

- Added license scheme Using RSD(Real Scan Device)

Version 3.0.0.14

- UFScanner - Added function which get number of Minutiae
  - UFS_GetFeatureNumber
- UFMatcher - Added parameter for 360-degree matching.
  - UFM_PARAM_AUTO_ROTATE (If set 1 at SetParameter support 360-degree matching)

Version 3.0.0.13

- Support MAC address type license

Version 3.0.0.12

- UFMatcher - UFM_RotateTemplate function supports SIF type

Version 3.0.0.11

- Modify SFR200's extract algorithm parameter for template compatibility between SFR200 and OP2/OP3

Version 3.0.0.8

- Error correction about UFScanner.dll SFR300 Ver.2's Logon process

Version 3.0.0.7

- Modify UFDatabase.mdb file problem
- UFDatabase.dll - Add error syntax when DB open

Version 3.0.0.5

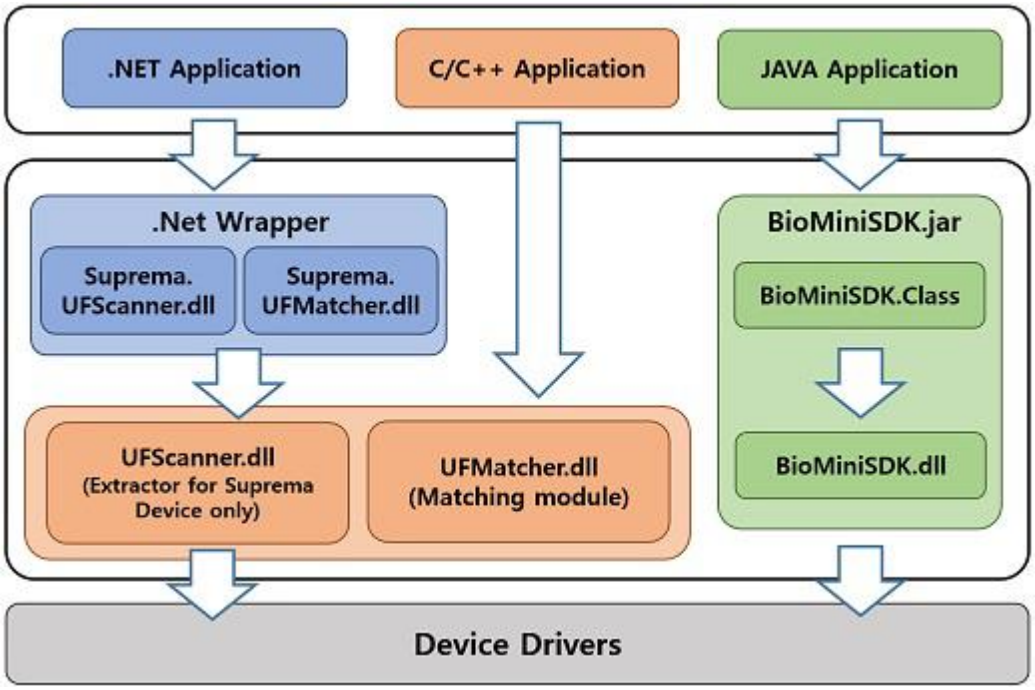
- Modify maximum resolution 640 x 640
- Change UFScanner_IZZIX.dll to latest versions(1.1.6.22)

Version 3.0.0.4

- Modify Identify function Fast Mode and improve speed: About 20000 matches/sec

1.3. OVERVIEW

SDK Structure



BioMini SDK contains UFScanner module to control a device and UFMatcher module to perform verification and identification. Also, it provides Suprema.UFScanner.dll, Suprema.UFMatcher.dll for .NET development and BioMIniSDK.jar for JAVA development.

Features

- Gives various application samples to developers and is easy to use

- Supports fingerprint minutiae formats below
  - Suprema: Proprietary fingerprint minutiae format of Suprema ID
  - ANSI378: Finger Minutiae Format for Data Exchange (ANSI-INCITS 378-2004)
  - ISO19794-2: Finger Minutiae Data (ISO/IEC 19794-2:2005)
- Provides low-level APIs for image capture, fingerprint feature extracting and matching

1.4. INSTALLATION

System Requirements

BioMini USB Scanners capture a fingerprint image and digitize it to an 8bit gray-scale image at 500ppi resolution. The host system receives the image through its USB for subsequent processing. All BioMini Scanners, except for those based on HID USB communication, are supported in this SDK. The following are the system requirements for BioMini USB Scanners.

- Operating System
  - Windows 7 / 8 / 10, Windows Server 2003 / 2008 R2 / 2012 / 2016
- Hardware
  - Core 2 Duo 2.4GHz
  - 1GB or more RAM
  - USB port 2.0 or higher
- Development Tools
  - Visual Studio 9.0 (2008) or higher
  - .NET Framework 3.5
  - JAVA SDK 1.4 or higher using JNI (Java Native Interface)

Package Files

After installing BioMini SDK for Windows, the following required files will be copied to "C:\Program Files\Suprema ID\BioMini

bin\cert directory

localhost_cert.crt	Certificate for localhost URL
localhost_cert.key	key for the same

bin\html directory

array.generics.min.js	Opensource library to support array functionality used in sample javascript
BiominiWebAgent.js	Javascript file to utilize Web-agent APIs (replaceable)
favicon.ico	Icon for localhost URL
index.html	Example page for web-api connected with web-agent
jquery.min.js	jQuery library for sample javascript file

bin\java directory

BioMiniSDK.jar	BioMini SDK class library for Java
ImageSDK.jar	Image SDK class library for Java

bin directory

BioMini library for 32-bit platform

activate_lic.exe	utility to activate the license
AgentCtrl.exe	Web-Agent Control UI
BioMini_DemoCS.exe	CSharp sample execution file
BioMini_DemoVBNET.exe	VB.net sample execution file
BioMini_DemoVC.exe	CPP sample execution file
BioMiniWebAgent.exe	Web-Agent Control console application
CertMgr.Exe	Certificate manage tool
D.LC	Essential component of diagnosis tool
demoBioMini.jar	Java sample
demoBioMini_Java.bat	batch file to execute Java sample
demoImageSDK.jar	Java Image SDK sample
demoImageSDK_Java.bat	batch file to execute Java Image SDK sample
DiagnosisTool.exe	utility to generate log files which have the device conditions
hasp_winodows.dll	library file for the license
libeay32.dll	library file for security
list_uuids.exe	utility to generate UUID.txt
MultiScannerDemo.exe	Multi scanner demo using VC++ 9.0
NFIQ2.dll	NIST NFIQ2 definition library file
register_enrollui.bat	batch file to register IEnrollUI.dll
ssleay32.dll	library file for security
Suprema.UFMatcher.dll	UFMatcher.dll .Net wrapper
Suprema.UFScanner.dll	UFScanner.dll .Net wrapper
UFMatcher.dll	Fingerprint template matching library
UFScanner.dll	Fingerprint scanning library

bin\x64 directory

BioMini library for 64-bit platform

AgentCtrl_x64.exe	Web-Agent Control UI
BioMini_DemoCS.exe	CSharp sample execution file
BioMini_DemoVBNET.exe	VB.net sample execution file
BioMini_DemoVC.exe	CPP sample execution file
BiominiWebAgent_x64.exe	Web-Agent Control console application
hasp_winodows_x64.dll	library file for the license
libeay32.dll	library file for security
NFIQ2.dll	NIST NFIQ2 definition library file
ssleay32.dll	library file for security
Suprema.UFMatcher.dll	UFMatcher.dll .Net wrapper
Suprema.UFScanner.dll	UFScanner.dll .Net wrapper
UFMatcher.dll	Fingerprint template matching library
UFScanner.dll	Fingerprint scanning library

docs\ directory

Biomini_Web_Agent_TS_Guide.pdf
BioMini SDK Programming Manual for Windows.pdf
BioMini SDK Quick Guide for Windows.pdf

include\ directory

UFMatcher.h
UFScanner.h
UFScannerDef.h

Install\ directory

drivers\SFR_Driver(unified)\Sup_Fingerprint_Driver_v2.2.1.exe

Redistribution Package\
vc2008redist_x64.exe
vc2008redist_x86.exe
vc2013redist_x64.exe
vc2013redist_x86.exe
vc2015redist_x64.exe
vc2015redist_x86.exe

lib\ directory

UFScanner.lib
UFMatcher.lib

x64\UFScanner.lib
x64\UFMatcher.lib

samples\ directory

Demo programs' source code project files

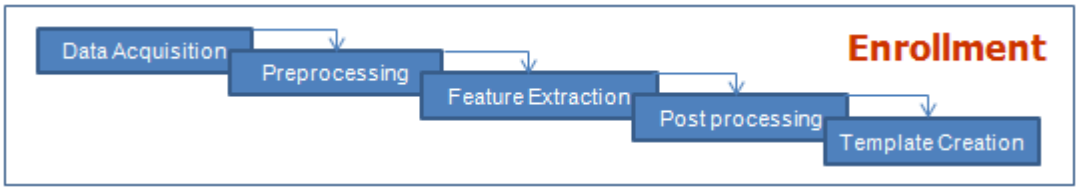
Java\demoBioMini
BioMini_DemoCS
BioMini_DemoVBNET
BioMini_DemoVC
MultiScannerDemoVC

Suprema_PC_SDK_3.9.1_Uninstall.exe

Tip! Please refer BioMini SDK Quick Guide for Windows.pdf when you install a driver and SDK package installer.

1.5. TUTORIAL

Enrollment



An enrollment application:

- Captures multiple fingerprints for fingers more than two fingers at least from a scanner.
- Checks image quality to ensure that a good quality image is obtained.
- Extracts the fingerprint minutiae.
- Saves the fingerprint images and / or minutiae in a database.

During the enrollment process, one or more fingers are scanned for each person. We recommend that you enroll two fingers at least. Because in the event of an accident or injury to one finger, another enrolled finger can be used to identify the individual.

The enrollment application needs to perform the following steps to enroll a finger from a user:

Workflow

- UFS_Init() — To initialize the scanner
- UFS_GetScannerHandle() — To get a scanner's handle
- UFS_Setparameter() — To set up the scanner's parameters
- UFS_SetTemplateType() — To set up the type of the template
- UFS_CaptureSingleImage() — To start to acquire the fingerprint image
- UFS_Uninit() — To release the scanner

Examples

- Preliminaries

```
//Add Suprema ID UFScanner lib (lib\UFScanner.lib) to the project.
//And add this statement to declare the header.
#include "UFScanner.h"

//Default valus is 1024 bytes
#define  MAX_TEMPLATE_SIZE        1024
```

- Initialize the scanner and check the number of scanners

```
UFS_STATUS ufs_res;
Int nScannerNumber;

//Initialize the scanner module
ufs_res = UFS_Init();

//Always check status return codes after running SDK functions.
//Meaning of status return code can be retrieved using UFS_GetErrorString()
//In this tutorial, we omit error check codes.

//Check the number of scanners
//If the number of scanners is under one, that means there is no scanner in a system.
ufs_res = UFS_GetScannerNumber(&nScannerNumber);
```

- Get the first scanner's handle

```
UFS_STATUS ufs_res;
HUFScanner hScanner = NULL;
int nCurScannerIndex;

//Get the first scanner's handle (If nCurScannerIndex is 0, it means the first scanner.)
ufs_res = UFS_GetScannerHandle(nCurScannerIndex, &hScanner);
```

- Set parameters

```
UFS_STATUS ufs_res;
int nValue;

//Set timeout for capturing images to 5 seconds.
//hScanner is a handle which comes from UFS_GetScannerHandle()
nValue = 5000;
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT, &nValue);

//Set template size to 1024 bytes
nValue = MAX_TEMPLATE_SIZE;
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TEMPLATE_SIZE, &nValue);

//Set not to detect core when extracting template
nValue = FALSE;
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_DETECT_CORE, &nValue);
```

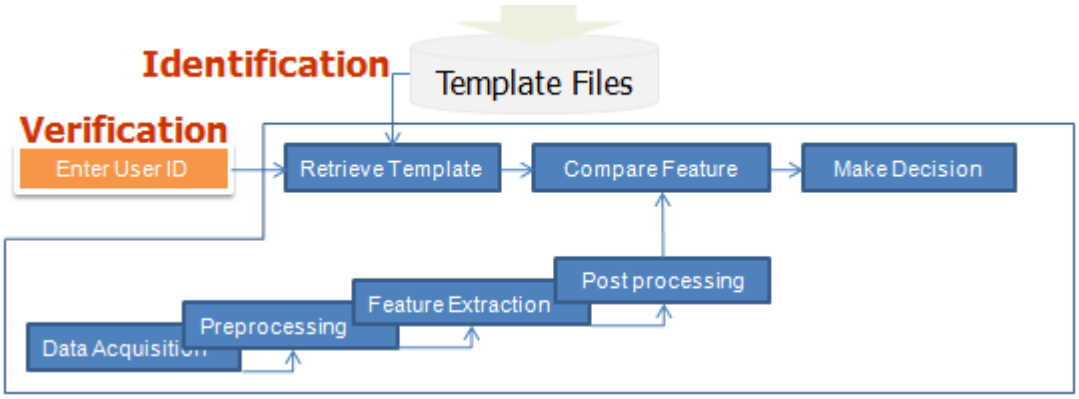
- Capture an image and extract a template

```
UFS_STATUS ufs_res;
unsigned char aTemplate[MAX_TEMPLATE_SIZE];
int nTemplateSize;
int nEnrollQuality;

//Clear capturing buffer
//hScanner is a handle which comes from UFS_GetScannerHandle()
ufs_res = UFS_ClearCaptureImageBuffer(hScanner);

//Capture a single image
//If the function is fail capturing image, iterate the capturing routine or show an error message.
ufs_res = UFS_CaptureSingleImage(hScanner);

//Extract template from a captured image
//If extraction is succeed, check nEnrollQuality is above predefined quality threshold.
ufs_res = UFS_ExtractEx(hScanner, MAX_TEMPLATE_SIZE, aTemplate, &nTemplateSize,
&nEnrollQuality);
```

- Release the scanner

```
UFS_STATUS ufs_res;

//Uninitialize the scanner module
ufs_res = UFS_Uninit();
```

Verification



Fingerprint recognition involves operation:

Verification – Comparing a fingerprint against a specific user's enrolled fingerprint(s) to verify a specific person's identity (e.g. when the user types their name and then uses a fingerprint rather than a password).

Workflow

- UFS_Init() To initialize the scanner
- UFS_GetScannerHandle() To get a scanner's handle
- UFS_Setparameter() To set up the scanner's parameters
- UFS_SetTemplateType() To set up the type of the template
- UFS_CaptureSingleImage() To start to acquire the fingerprint image
- UFS_Extract() To extract the captured image to template
- UFM_Create() To create a Matcher for matching
- UFM_Verify() To compare it to a selected template in database
- UFM_Delete() To close a Matcher
- UFS_Uninit() To release the scanner

Examples

- Preliminaries

```
//Add Suprema ID UFMather lib (lib\UFMatcher.lib) to the project.
//And add this statement to declare the header.
#include "UFMatcher.h"

//Default valus is 1024 bytes
#define  MAX_TEMPLATE_SIZE        1024
```

- Create a matcher

```
UFM_STATUS ufm_res;
HUFMatcher hMatcher = NULL;

//Always check status return codes after running SDK functions.
//Meaning of status return code can be retrieved using UFM_GetErrorString()
//In this tutorial, we omit error check codes.

//Create a matcher
ufm_res = UFM_Create(&hMatcher);
```

- Set parameters

```
UFM_STATUS ufm_res;
int nValue;

//Set Security level to 4
nValue = 4;
ufs_res = UFS_SetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL, &nValue);

//Set FastMode on
nValue = 1;
ufm_res = UFS_SetParameter(hMatcher, UFS_PARAM_FAST_MODE, &nValue);
```

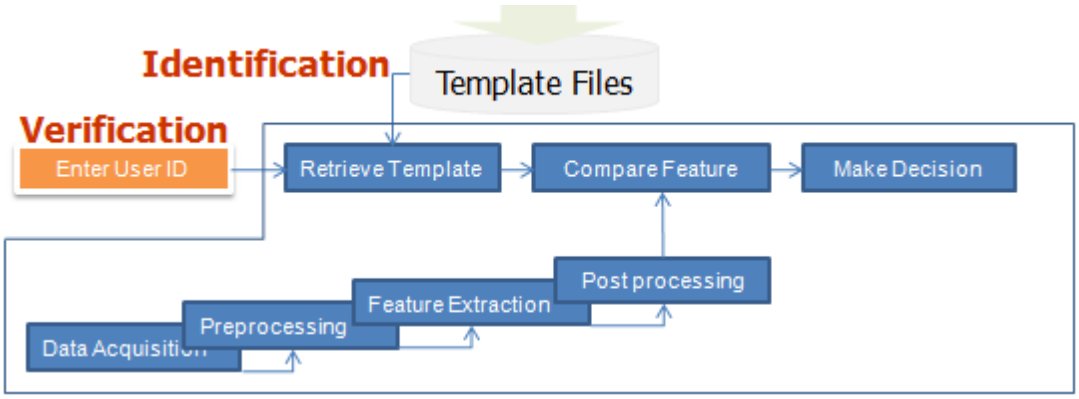
- Capture an image, extract a template and verify a fingerprint

```
UFS_STATUS ufs_res;
unsigned char aTemplate[MAX_TEMPLATE_SIZE];
int nTemplateSize;
int nEnrollQuality;

//Clear capturing buffer
//hScanner is a handle which comes from UFS_GetScannerHandle()
ufs_res = UFS_ClearCaptureImageBuffer(hScanner);

//Capture a single image
//If the function is fail capturing image, iterate the capturing routine or show an error message.
ufs_res = UFS_CaptureSingleImage(hScanner);

//Extract template from a captured image
//If extraction is succeed, check nEnrollQuality is above predefined quality threshold.
ufs_res = UFS_ExtractEx(hScanner, MAX_TEMPLATE_SIZE, aTemplate, &nTemplateSize,
&nEnrollQuality);

//Verify an enrolled fingerprint with the captured fingerprint
ufm_res = UFM_Verify(hMatcher, Template, TemplateSize, enrolledTemplate,
enrolledTemplateSize, &bVerifySucceed);
if (ufm_res != UFM_OK)
{
        //To add error handling codes
}

if (bVerifySucceed)
{
        //Verification success
}
else
{
        //Verification failure
}
```

- Delete a matcher

```
UFM_STATUS ufm_res;

//Delete a matcher
ufm_res = UFM_Delete(&hMatcher);
```

Identification



Fingerprint recognition involves operation:

Identification – Comparing a fingerprint against the database of enrolled fingerprints and confirming that the fingerprint is enrolled (e.g. To open a door, we should compare one's fingerprint with many authorized users).

Workflow

- UFS_Init() To initialize the scanner
- UFS_GetScannerHandle() To get a scanner's handle
- UFS_Setparameter() To set up the scanner's parameters
- UFS_SetTemplateType() To set up the type of the template
- UFS_CaptureSingleImage() To start to acquire the fingerprint image
- UFS_Extract() To extract the captured image to template
- UFM_Create() To create a Matcher for matching
- UFM_Identify() To compare it to a selected template in database
- UFM_Delete() To close a Matcher
- UFS_Uninit() To release the scanner

Examples

- Preliminaries

```
//Add Suprema ID UFMather lib (lib\UFMatcher.lib) to the project.
//And add this statement to declare the header.
#include "UFMatcher.h"

//Default valus is 1024 bytes
#define  MAX_TEMPLATE_SIZE       1024

//Set maximum template number to 50 (this depends on an application).
#define  MAX_TEMPLATE_NUM        50
```

- Create a matcher

```
UFM_STATUS ufm_res;
HUFMatcher hMatcher = NULL;

//Always check status return codes after running SDK functions.
//Meaning of status return code can be retrieved using UFM_GetErrorString()
//In this tutorial, we omit error check codes.

//Create a matcher
ufm_res = UFM_Create(&hMatcher);
```

- Set parameters

```
UFM_STATUS ufm_res;
int nValue;

//Set Security level to 4
nValue = 4;
ufs_res = UFS_SetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL, &nValue);

//Set FastMode on
nValue = 1;
ufm_res = UFS_SetParameter(hMatcher, UFS_PARAM_FAST_MODE, &nValue);
```

- Identify

```
UFM_STATUS ufm_res;
unsigned char Template1[MAX_TEMPLATE_SIZE];
int Template1Size;
unsigned char* Template2Array[MAX_TEMPLATE_NUM];
int Template2SizeArray[MAX_TEMPLATE_NUM];
int Template2Num;
int nMatchIndex;

// Allocate Template2Array
for (i = 0; i < MAX_TEMPLATE_NUM; i++)
{
        Template2Array[i] = (unsigned char*)malloc(MAX_TEMPLATE_SIZE);
        memset(Template2Array[i], , MAX_TEMPLATE_SIZE);
}

// Get Template1 from a scanner or an image or a database
// Get Template2Array from a scanner or an image or a database
// Identify Template1 from Template2Array, set timeout to 5 seconds
ufm_res = UFM_Identify(hMatcher, Template1, Template1Size, Template2Array,
Template2SizeArray, Template2Num, 5000, &nMatchIndex);
if (ufm_res != UFM_OK)
{
        // To add error handling codes
}
else
{
        if (nMatchIndex != -1)
        {
                // Identification success
        }
        else
        {
                // Identification failure
        }
}
// Free Template2Array
for (i = 0; i < MAX_TEMPLATE_NUM; i++)
{
        free(Template2Array[i]);
}
```

- Delete a matcher

```
UFM_STATUS ufm_res;

//Delete a matcher
ufm_res = UFM_Delete(&hMatcher);
```

Chapter 2. PROGRAMMING IN C/C++

2.1. SETTING ENVIRONMENT

This section will explain how to import BioMini Libraries (UFScanner.dll, UFMatcher.dll) on Microsoft Visual Studio 2013.

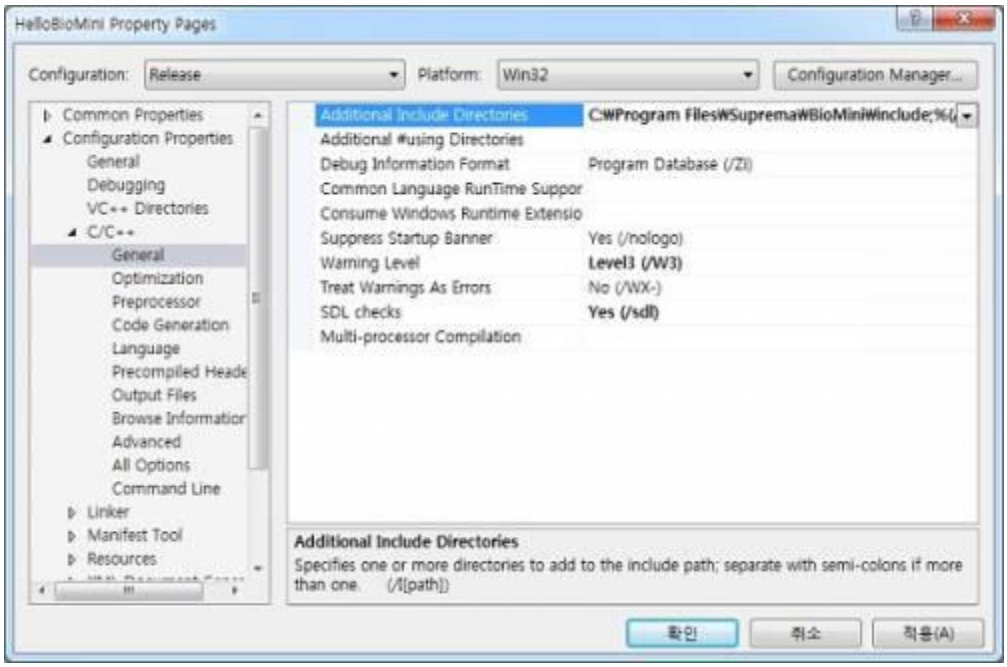
Create a new project

Create new MFC dialog-based application project.

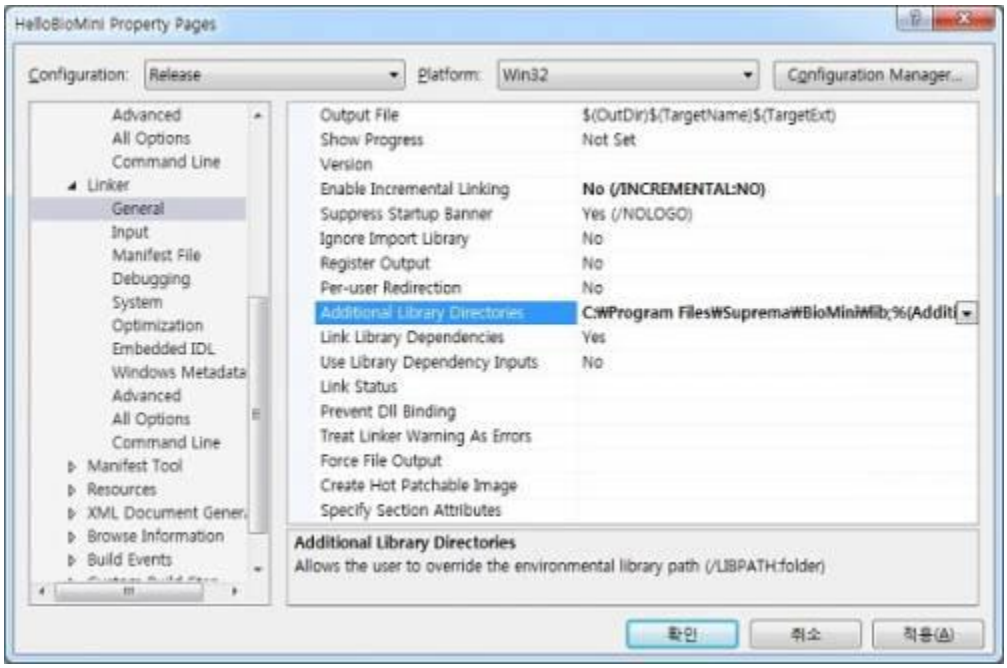
Select [Dialog based] for application type.

Configure project properties

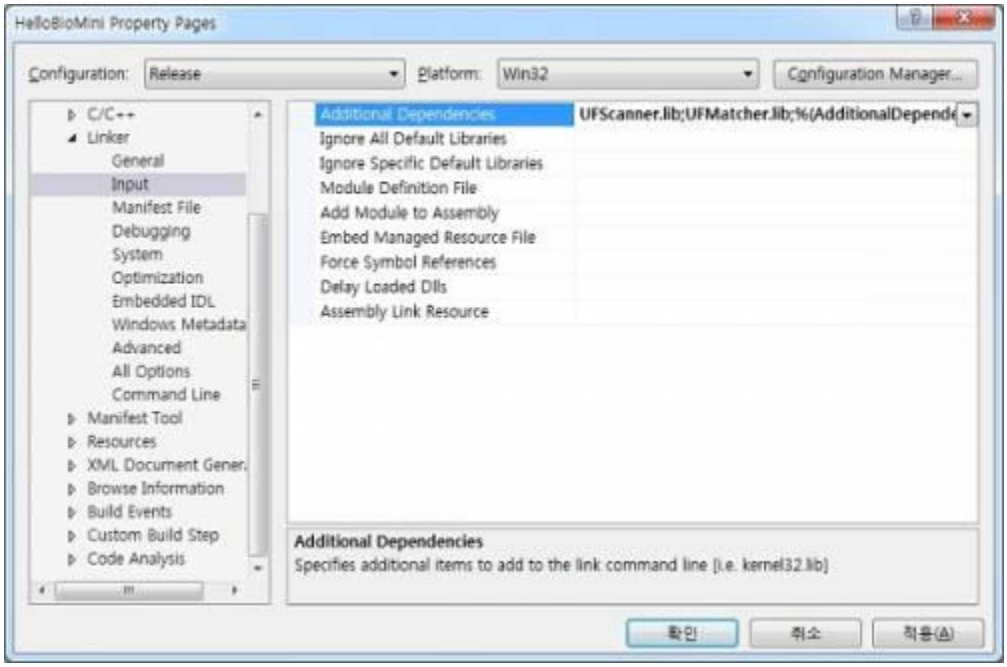
Add include files: Add "%SDKDIR%include\" folder at "Project Property Pages > C/C++ > General > Additional Include Directories". (ex. C:\Program Files(x86)\Suprema\BioMini\include\)



Add lib files: Add "%SDKDIR%lib\" folder at "Project Property Pages > Linker > General > Additional Include Directories". (ex. C:\Program Files(x86)\Suprema\BioMini\lib\)

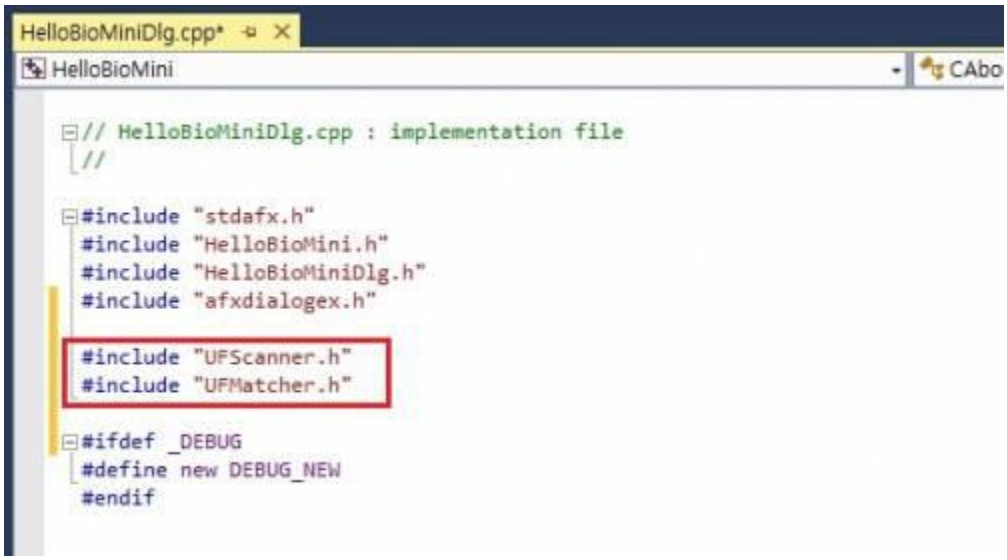


Add ‘additional dependencies’: Add “UFScanner.lib, UFMatcher.lib” at “Project Property Pages > Linker > Input > Additional Dependencies”.



Add a basic code

Declare include header files: Add “UFScanner.h, UFMatcher.h” to “<ProjectName>Dlg.cpp” source.



Add initializing code at OnInitDialog().

```
// TODO: Add extra initialization here
UFS_STATUS ufs_res;
int nScannerNumber;

ufs_res = UFS_Init();
if (ufs_res == UFS_OK)
{
	//UFS_Init success
}
else
{
	//UFS_Init failure
}

ufs_res = UFS_GetScannerNumber(&nScannerNumber);
if (ufs_res == UFS_OK)
{
	AddMessage("UFS_GetScannerNumber: %d\r\n", nScannerNumber);
}
else
{
	//UFS_GetScannerNumber failure
}
```

Add releasing code at OnClose()

```
Void CHelloBioMiniDlg::OnClose()
{
	//TODO: Add your message handler code here and/or call default
	UFS_Uninit();
	CDialogEx::OnClose();
}
```

Check output files

Compile the project and copy UFScanner.dll and UFMatcher.dll at $(SolutionDir)Debug\

Can check message as below by running HelloBioMini.exe after connecting the device.

2.2. API REFERENCE

UFScanner Functions

UFS_Init

Initializes a UFScanner module.

Syntax

```
UFS_STATUS UFS_API UFS_Init()
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS    usf_res;

ufs_res = UFS_Init();
if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_Update

Enforces a UFScanner module to update the connection state of scanners.

Syntax

```
UFS_STATUS UFS_API UFS_Update()
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS    usf_res;

ufs_res = UFS_Update();
if (ufs_res == UFS_OK)
{
        //UFS_Update success
}
else
{
        //UFS_Update failure
        //Can use UFS_GetErrorString function to show an error string.
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_Uninit

Un-initializes a UFScanner module.

Syntax

```
UFS_STATUS UFS_API UFS_Uninit()
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS        usf_res;

ufs_res = UFS_Uninit();
if (ufs_res == UFS_OK)
{
        //UFS_Uninit success
}
else
{
        //UFS_Uninit failure
        //Can use UFS_GetErrorString function to show an error string.
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SetScannerCallback

Registers the scanner callback function.

Syntax

```
UFS_STATUS UFS_API UFS_SetScannerCallback(
                                    UFS_SCANNER_PROC* pScannerProc,
                                    void* pParam)
```

Parameters

- pScannerProc [in]: Handle to the UFS_SCANNER_PROC function which receives scanner events
- pParam [in]: Pointer to the scanner callback data which will be transmitted with a scanner callback event

Return value

UFS_STATUS

Remarks

Examples

```
# define UFS_CALLBACK __stdcall

int UFS_CALLBACK ScannerProc(const char* szScannerID, int bSensorOn, void* pParam)
{
	CBiomini_DemoDlg* pDlg = (CBiomini_DemoDlg*)pParam;

	// Can control UI with WM message.

	return 1;
}

-----------------------------------------------------------------------------------------------------------
UFS_STATUS	usf_res;

ufs_res = UFS_SetScannerCallback(ScannerProc, (void*)this);
if (ufs_res == UFS_OK)
{
	//UFS_SetScannerCallback success
}
else
{
	//UFS_SetScannerCallback failure
	//Can use UFS_GetErrorString function to show an error string.
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_RemoveScannerCallback

Removes the registered scanner callback function.

Syntax

```
UFS_STATUS UFS_API UFS_RemoveScannerCallback ()
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS        usf_res;

ufs_res = UFS_RemoveScannerCallback();
if (ufs_res == UFS_OK)
{
        //UFS_RemoveScannerCallback success
}
else
{
        //UFS_ RemoveScannerCallback failure
        //Can use UFS_GetErrorString function to show an error string.
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerNumber

Gets the number of scanners.

Syntax

```
UFS_STATUS UFS_API UFS_GetScannerNumber(int* pnScannerNumber)
```

Parameters

- pnScannerNumber [out]: Receive the number of scanners

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
int             nScannerNumber;

ufs_res = UFS_GetScannerNumber(&nScannerNumber);
if (ufs_res == UFS_OK)
{
        //UFS_GetScannerNumber success
}
else
{
        //UFS_ GetScannerNumber failure
        //Can use UFS_GetErrorString function to show an error string.
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerHandle

Gets the scanner handle using a scanner index.

Syntax

```
UFS_STATUS UFS_API UFS_GetScannerHandle(int nScannerIndex,
                                        HUFScanner* phScanner)
```

Parameters

- nScannerIndex [in]: Scanner index (0 ~ number of scanners -1)
- phScanner [out]: Pointer to handle of the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
int             nScannerIndex;
HUFScanner      hScanner = NULL;

//UFS_GetScannerHandle can retrieve the number of scanners.
ufs_res = UFS_GetScannerHandle(nScannerIndex, &hScanner);
if (ufs_res == UFS_OK)
{
        //UFS_GetScannerHandle success
}
else
{
        //UFS_ GetScannerHandle failure
        //Can use UFS_GetErrorString function to show an error string.
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerHandleByID

Gets the scanner handle using a scanner ID.

Syntax

```
UFS_STATUS UFS_API UFS_GetScannerHandleByID (
                                const char* szScannerID,
                                HUFScanner* phScanner)
```

Parameters

- szScannerID [in]: The scanner ID
- phScanner [out]: Pointer to handle of the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
char            strID[64];
HUFScanner      hScanner = NULL;

//Assign the scanner ID to strID and call UFS_ GetScannerHandleByID to retrieve the scanner ID.
ufs_res = UFS_GetScannerHandleByID(strID, &hScanner);
if (ufs_res == UFS_OK)
{
        //UFS_ GetScannerHandleByID success
}
else
{
        //UFS_ GetScannerHandleByID failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerIndex

Initializes a UFScanner module.

Syntax

```
UFS_STATUS UFS_API UFS_GetScannerIndex (HUFScanner* phScanner,
                                        int* pnScannerIndex)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pnScannerIndex [out]: Retrieve scanner index of specified scanner handle

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
HUFScanner      hScanner = NULL;
int             nScannerIndex;

//Get a scanner's handle and call UFS_GetScannersIndex
ufs_res = UFS_GetScannersIndex();
if (ufs_res == UFS_OK)
{
        //UFS_ GetScannersIndex success
}
else
{
        //UFS_ GetScannersIndex failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerID

Gets scanner ID assigned to scanner

Syntax

```
UFS_STATUS UFS_API UFS_GetScannerID(HUFScanner hScanner,
                                    char* szScannerID)
```

Parameters

- hScanner [in]: Handle to the scanner object
- szScannerID [out]: Retrieve scanner ID of specified scanner handle; Scanner ID has maximum 32 characters. szScannerID must be allocated in user's applications and allocated size must be larger than 33 bytes for considering null character in 33th byte position.

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
HUFScanner      hScanner = NULL;
char*           strID[64];
//should be larger than 33bytes

ufs_res = UFS_GetScannerID(hScanner, strID);
if (ufs_res == UFS_OK)
{
        //UFS_GetScannerID success
}
else
{
        //UFS_ GetScannerID failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerType

Gets the scanner type that is assigned to the scanner handle.

Syntax

```
UFS_STATUS UFS_API UFS_GetScannerType(UFScanner hScanner,
                                      int* pnScannerType)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pnScannerType [out]: Retrieve one of the scanner types

Scanner Type	Code	Description
UFS_SCANNER_TYPE_SFR200	1001	Suprema SFR200
UFS_SCANNER_TYPE_SFR300	1002	Suprema SFR300-S
UFS_SCANNER_TYPE_SFR300v2	1003	Suprema SFR300v2, SFR400
UFS_SCANNER_TYPE_SFR500	1004	BioMini Plus
UFS_SCANNER_TYPE_SFR600	1005	BioMini Slim
UFS_SCANNER_TYPE_SFR410	1006	BioMini OC4
UFS_SCANNER_TYPE_SFR550	1007	BioMini Plus 2
UFS_SCANNER_TYPE_SFR700	1008	BioMini Slim 2
UFS_SCANNER_TYPE_BMS3	1012	BioMini Slim 3

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
UFScanner       hScanner;
int             nScannerType;

ufs_res = UFS_GetScannterType(hScanner, &nScannerType);
if (ufs_res == UFS_OK)
{
        //UFS_GetScannerType success
}
else
{
        //UFS_ GetScannerType failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetParameter

Gets parameter value of UFScanner module.

Syntax

```
UFS_STATUS UFS_API UFS_GetParameter(HUFScanner hScanner,
                                    int nParam,
                                    void* pValue)
```

Parameters

- hScanner [in]: Handle to the scanner object
- nParam [in]: Parameter type; one of parameters

Parameter	Code	Description	Default Value
UFS_PARAM_TIMEOUT	201	Timeout (millisecond unit) (0: infinite)	5000
UFS_PARAM_BRIGHTNESS	202	Brightness (0~255): Higher value means darker image. * Supported Device: BioMini (SFR400, SFR410), BioMini Plus(SFU-500)	100
UFS_PARAM_SENSITIVITY	203	Sensitivity (0~7): Higher value means more sensitive	4
UFS_PARAM_SERIAL	204	Serial (get only)	-
UFS_PARAM_SDK_VERSION	210	SDK Version (get only)	
UFS_PARAM_SDK_COPYRIGHT	211	SDK Copyright (get only)	
UFS_PARAM_SCANNING_MODE	220	Image size of BioMini Plus 2 (0: 288x340[pixels], 1: 315x354[pixels])	0
UFS_PARAM_DETECT_CORE	301	Detect core (0: not use core, 1: use core)	0
UFS_PARAM_TEMPLATE_SIZE	302	Template size (byte unit) (256~1024, 32bytes step size)	1024
UFS_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFS_PARAM_DETECT_FAKE	312	Use live Finger Detection (0: not use LFD, 1~5: use LFD) Higher value means more strong to defend a fake finger * Supported Device: BioMini Slim (SFU-S20)	0
UFS_PARAM_LFD_TYPE	313	LFD operation options (0: UFS_LFD_TYPE_DEFAULT, 1: UFS_LFD_TYPE_ADVANCED) * Supported device: BioMini Slim (SFU-S20)	-
UFS_PARAM_LFD_FILE	314	Path of current LFD Engine file * Supported device: BioMini Plus 2 (SFR-550), BioMini Slim (SFU-S20)	-
UFS_PARAM_LFD_SCORE	317	LFD score. * Supported device: BioMini Slim 2, BioMini Slim 3	
UFS_PARAM_LANGUAGE	401	Language selection at runtime of EnrollUI	-
UFS_PARAM_FPQUALITY_MODE	402	Quality score type UFS_NQS_MODE_DEFAULT (0): Suprema quality score, Quality value ranges from 0(Poorest) to 100(Highest) UFS_NQS_MODE_NFIQ_PERCENTILE (1): NFIQ 1.0 quality score, 5 levels of quality value between 20(Poorest) and 100(Highest)	0

		UFS_NQS_MODE_NFIQ	2	NFIQ 1.0 quality score, 5 levels of quality value between 5(Poorest) and 1(Highest)	
		UFS_NQS_MODE_NFIQ2	3	NFIQ 2.0 quality score, Quality value ranges from 0(Poorest) to 100(Highest)	
UFS_PARAM_NFIQ2_FILE	406	Path of the NFIQ2.0 Engine file			N/A

- *pValue* [out]: Receives parameter value of specified parameter type. pValue must point to adequate storage type matched to parameter type.

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nValue;

// After getting hScanner handle
// Get timeout
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_TIMEOUT, &nValue);
if (ufs_res == UFS_OK)
{
        //UFS_GetParameter success
}
else
{
        //UFS_ GetParameter failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SetParameter

Sets parameter value of UFScanner module.

Syntax

```
UFS_STATUS UFS_API UFS_SetParameter(HUFScanner hScanner,
                                    int nParam,
                                    void* pValue)
```

Parameters

- hScanner [in]: Handle to the scanner object
- nParam [in]: Parameter type; one of parameters

Parameter	Code	Description	Default Value
UFS_PARAM_TIMEOUT	201	Timeout (millisecond unit) (0: infinite)	5000
UFS_PARAM_BRIGHTNESS	202	Brightness (0~255): Higher value means darker image. * Supported Device: BioMini (SFR400, SFR410), BioMini Plus(SFU-500)	100
UFS_PARAM_SENSITIVITY	203	Sensitivity (0~7): Higher value means more sensitive	4
UFS_PARAM_SCANNING_MODE	220	Adjust the image size of BioMini Plus 2 (0: 288x340[pixels], 1: 315x354[pixels])	0
UFS_PARAM_DETECT_CORE	301	Detect core (0: not use core, 1: use core)	0
UFS_PARAM_TEMPLATE_SIZE	302	Template size (byte unit) (256~1024, 32bytes step size)	1024
UFS_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFS_PARAM_DETECT_FAKE	312	Use live Finger Detection (0: not use LFD, 1~5: use LFD) Higher value means more strong to defend a fake finger * Supported Device: BioMini Slim (SFU-S20)	0
UFS_PARAM_LFD_TYPE	313	LFD operation options (0: UFS_LFD_TYPE_DEFAULT, 1: UFS_LFD_TYPE_ADVANCED LFD checking level is upgrade so it can be expected improved defense performance against to some counterfeit fingerprints.) * Supported device: BioMini Slim (SFU-S20)	0
UFS_PARAM_LFD_FILE	314	Specify the path of the engine file for upgrading the LFD Engine. Since the default engine is built into the SDK, you do not need to call it unless it is for upgrade purposes. * Supported device: BioMini Plus 2 (SFR-550), BioMini Slim (SFU-S20)	-
UFS_PARAM_LANGUAGE	401	Language selection at runtime of EnrollUI	-
UFS_PARAM_FPQUALITY_MODE	402	Quality score type UFS_NQS_MODE_DEFAULT (0): Suprema quality score, Quality value ranges from 0(Poorest) to 100(Highest) UFS_NQS_MODE_NFIQ_PERCENTILE (1): NFIQ 1.0 quality score, 5 levels of quality value between 20(Poorest) and 100(Highest) UFS_NQS_MODE_NFIQ (2): NFIQ 1.0 quality score, 5 levels of quality value between 5(Poorest) and 1(Highest) UFS_NQS_MODE_NFIQ2 (3): NFIQ 2.0 quality score, Quality value ranges	0

		from 0(Poorest) to 100(Highest)	
UFS_PARAM_NFIQ2_FILE	406	Specify the path of the NFIQ2.0 Engine file The NFIQ2.0 library file in the BioMini PC SDK package is "NFIQ2.dll". (If not set, you cannot use NFIQ2.0.)	N/A

- pValue [in]: Pointer to parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nValue;

// After getting hScanner handle
// Set timeout
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT, &nValue);
if (ufs_res == UFS_OK)
{
        //UFS_SetParameter success
}
else
{
        //UFS_ SetParameter failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_IsSensorOn

Checks whether a scanner is connected or not.

Syntax

```
UFS_STATUS UFS_API UFS_IsSensorOn(HUFScanner hScanner,
                                  int* pbSensorOn)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pbSensorOn [out]: Receive the status of specified scanner object;
  1: the scanner is connected. / 0: the scanner is disconnected.

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             bSensorOn;

// After getting hScanner handle
ufs_res = UFS_IsSensorOn(hScanner, &bSensorOn);
if (ufs_res == UFS_OK)
{
        //UFS_IsSensorOn success
}
else
{
        //UFS_ IsSensorOn failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_IsFingerOn

Checks whether a finger is placed on a scanner or not.

Syntax

```
UFS_STATUS UFS_API UFS_IsFingerOn(HUFScanner hScanner,
                                  int* pbFingerOn)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pbFingerOn [out]: Checks if a finger is placed on the specified scanner;
  1: a finger is on the scanner / 0: a finger is not on the scanner

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             bFingerOn;

// After getting hScanner handle
ufs_res = UFS_IsFingerOn(hScanner, &bFingerOn);
if (ufs_res == UFS_OK)
{
        //UFS_IsFingerOn success
}
else
{
        //UFS_ IsFingerOn failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_CaptureSingleImage

Captures single image. Captured image is stored to the internal buffer.

Syntax

```
UFS_STATUS UFS_API UFS_CaptureSingleImage(HUFScanner hScanner)
```

Parameters

- hScanner [in]: Handle to the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;

// After getting hScanner handle

ufs_res = UFS_CaptureSingleImage(hScanner);
if (ufs_res == UFS_OK)
{
        //UFS_CaptureSingleImage success
}
else
{
        //UFS_ CaptureSingleImage failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_StartCapturing

Starts capturing. The capture is going on until the specified time exceeds.

Syntax

```
UFS_STATUS UFS_API UFS_StartCapturing(HUFScanner hScanner,
                                      UFS_CAPTURE_PROC* pCaptureProc,
                                      void* pParam)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pCaptureProc [in]: Handle to the UFS_CAPTURE_PROC function which receives capture events
- pParam [in]: Pointer to the capture callback data which will be transmitted with a capture callback event

Return value

UFS_STATUS

Remarks

Examples

```
// Define capture procedure
int UFS_CALLBACK CaptureProc(HUFScanner hScanner, int bFingerOn, unsigned char* pImage,
                                    int nWidth, int nHeight, int nResolution, void* pParam)
{
        // ...
}

UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
void*           pParam;

// After getting hScanner handle

// Assign pParam, for example, application data
ufs_res = UFS_StartCapturing(hScanner, CaptureProc, pParam);
if (ufs_res == UFS_OK)
{
        // UFS_StartCapturing success
}
else
{
        // UFS_StartCapturing failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_StartAutoCapture

Starts the automatic capture. Currently this function is working for Suprema ID SFR-600 (BioMini Slim) only.

Syntax

```
UFS_STATUS UFS_API UFS_StartAutoCapture(HUFScanner hScanner,
                                        UFS_CAPTURE_PROC* pCaptureProc,
                                        void* pParam)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pCaptureProc [in]: Handle to the UFS_CAPTURE_PROC function which receives capture events
- pParam [in]: Pointer to the capture callback data which will be transmitted with a capture callback event

Return value

UFS_STATUS

Remarks

Examples

```
// Define capture procedure
int UFS_CALLBACK CaptureProc(HUFScanner hScanner, int bFingerOn, unsigned char* pImage,
                                        int nWidth, int nHeight, int nResolution, void* pParam)
{
        // ...
}

UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
void*           pParam;

// After getting hScanner handle

// Assign pParam, for example, application data
ufs_res = UFS_StartAutoCapture(hScanner, CaptureProc, pParam);
if (ufs_res == UFS_OK)
{
        // UFS_StartAutoCapture success
}
else
{
        // UFS_ StartAutoCapture failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_IsCapturing

Checks if the specified scanner is running to capture which started by UFS_CaptureSingleImage or UFS_StartCapturing.

Syntax

```
UFS_STATUS UFS_API UFS_IsCapturing(HUFScanner hScanner,
                                   int* pbCapturing)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pbCapturing [out]: Checks if the specified scanner is running capturing;
  1: the capture is running / 0: the capture is not running

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
HUFScanner      hScanner = NULL;
int             bCapturing;

// After getting hScanner handle
ufs_res = UFS_IsCapturing(hScanner, &bCapturing);
if (ufs_res == UFS_OK)
{
        //UFS_IsCapturing success
}
else
{
        //UFS_IsCapturing failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_STATUS

UFS_AbortCapturing

Aborts capturing which is started by UFS_CaptureSingleImage or UFS_StartCapturing.

Syntax

```
UFS_STATUS UFS_API UFS_AbortCapturing(HUFScanner hScanner)
```

Parameters

- hScanner [in]: Handle to the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
HUFScanner      hScanner = NULL;

// After getting hScanner handle

ufs_res = UFS_AbortCapturing(hScanner);
if (ufs_res == UFS_OK)
{
        //UFS_AbortCapturing success
}
else
{
        //UFS_AbortCapturing failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_STATUS

UFS_Extract

Extracts a template from the stored image buffer which is acquired using UFS_CaptureSingleImage or UFS_StartCapturing.

Syntax

```
UFS_STATUS UFS_API UFS_Extract(HUFScanner hScanner, unsigned char* pTemplate,
                               int* pnTemplateSize, int* pnEnrollQuality)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplate [out]: Pointer to the template array; The array must be allocated in advance.
- pnTemplateSize [out]: Receives the size (in bytes) of pTemplate
- pnEnrollQuality [out]: Receives the quality of enrollment; Quality value ranges from 1 to 100. Typically, this value should be above 30 for further processing such as enroll and matching. Especially in case of enrollment, the use of good quality image (above 50) is highly recommended.

Return value

UFS_STATUS

Remarks

Examples

```
// Template size can be controlled by using UFS_SetParameter function

// Default value is 1024 bytes
#define MAX_TEMPLATE_SIZE 1024

UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
unsigned char   Template[MAX_TEMPLATE_SIZE];
int             TemplateSize;
int             nEnrollQuality;

// After getting hScanner handle

ufs_res = UFS_Extract(hScanner, Template, &TemplateSize, &nEnrollQuality);
if (ufs_res == UFS_OK)
{
        // UFS_Extract success
}
else
{
        // UFS_Extract failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_STATUS

UFS_ExtractEx

Can extract a template from the stored image buffer which is acquired using UFS_CaptureSingleImage or USF_StartCapturing. This is an extended version of UFS_Extract function to accommodate large size template.

Syntax

```
UFS_STATUS UFS_API UFS_ExtractEx(HUFScanner hScanner, int* nBufferSize,
                                 unsigned char* pTemplate, int* pnTemplateSize, int* pnEnrollQuality);
```

Parameters

- hScanner [in: Handle to the scanner object
- nBufferSize [in: Template buffer size

Value	Meaning
MAX_TEMPLATE_SIZE	1024 and under

- pTemplate [in]: Pointer to the template array; The array must be allocated in advance.
- pnTemplateSize: Receive the size (in bytes) of pTemplate
- pnEnrollQuality: Receive the quality of enrollment;
  Quality value ranges from 1 to 100.Typically this value should be above 30 for further processing such as enroll and matching. Especially in case of enrollment, the use of good quality image (above 50) is highly recommended.

Return value

UFS_STATUS

Remarks

Examples

```
//Template size can be controlled by using UFS_SetParameter function.

//Default valus is 1024 bytes
#define  MAX_TEMPLATE_SIZE 1024

UFS_STATUS      usf_res;
HUFScanner      hScanner = NULL;
unsigned char   pTemplate[MAX_TEMPLATE_SIZE];
int             nTemplateSize;
int             nEnrollQuality;

ufs_res = UFS_ExtractEx(hScanner, MAX_TEMPLATE_SIZE, pTemplate, &TemplateSize, &nEnrollQuality);

if (ufs_res == UFS_OK)
{
        //UFS_ExtractEx success
}
else
{
        // UFS_ExtractEx failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_Extract
UFS_CaptureSingleImage
USF_StartCapturing
UFS_STATUS

UFS_SetEncryptionKey

Set encryption key.

Syntax

```
UFS_STATUS UFS_API UFS_SetEncryptionKey(HUFScanner hScanner,
                                        unsigned char* pKey)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pKey [out]: Pointer to the 32 bytes key array; default key is first byte is 1 and second to 32th byte are all 0.

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
unsigned char   UserKey[32];

// After getting hScanner handle
// Generate 32 byte encryption key to UserKey
ufs_res = UFS_SetEncryptionKey(hScanner, UserKey);
if (ufs_res == UFS_OK)
{
        // UFS_SetEncryptionKey success
}
else
{
        // UFS_SetEncryptionKey failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SetEncryptionKeyEx

Set encryption key for standard AES algorithm.

Syntax

```
UFS_STATUS UFS_API UFS_SetEncryptionKeyEx(HUFScanner hScanner,
                                          int nAESBlockMode,
                                          unsigned char* pKey,
                                          unsigned char* pInitialVector)
```

Parameters

- hScanner [in]: Handle to the scanner object
- nAESBlockMode [in]: Parameter type; one of AES algorithm

AES algorithm	Code	Description
UFS_ENC_MODE_DEFAULT	0	Old fashion
UFS_ENC_MODE_AES_ECB	1	Electronic Codebook
UFS_ENC_MODE_AES_CBC	2	Cipher Block Chaining
UFS_ENC_MODE_AES_PCBC	3	Propagating Cipher Block Chaining
UFS_ENC_MODE_AES_CRT	4	Cipher Feedback
UFS_ENC_MODE_AES_OFB	5	Output Feedback
UFS_ENC_MODE_AES_CFB	6	Counter

- pKey [out]: Pointer to the 32 bytes key array; default key is first byte is 1 and second to 32th byte are all 0.
- pInitialVector [in]: Pointer to the 16 bytes initial vector (set NULL if not needed). UFS_ENC_MODE_AES_ECB and UFS_ENC_MODE_DEFAULT modes don't need the initial vector.

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
unsigned char   UserKey[32];
unsigned char   InitialVector[16];
int             AESBlockMode;

// After getting hScanner handle
// Generate 32 byte encryption key to UserKey with InitialVector with AESBlockMode
ufs_res = UFS_SetEncryptionKeyEx(hScanner, AESBlockMode, UserKey, InitialVector);
if (ufs_res == UFS_OK)
{
        // UFS_SetEncryptionKeyEx success
}
else
{
        // UFS_SetEncryptionKeyEx failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_EncryptTemplate

Encryptes template.

Syntax

```
UFS_STATUS UFS_API UFS_EncryptTemplate(HUFScanner hScanner,
                                       unsigned char* pTemplateInput,
                                       int nTemplateInputSize,
                                       unsigned char* pTemplateOutput,
                                       int* pnTemplateOutputSize)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplateInput [in]: Pointer to input template data
- nTemplateInputSize [in]: Input template size
- pTemplateOutput [out]: Pointer to encrypted template data
- nTemplateOutputSize [in/out]: Inputs allocated size of encrypted template data; Receives output template size

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;

unsigned char   TemplateInput[MAX_TEMPLATE_SIZE];
unsigned char   TemplateOutput[MAX_TEMPLATE_SIZE];
int             TemplateInputSize;
int             TemplateOutputSize;

// After getting hScanner handle
// Get an input template to encrypt, TemplateInput and TemplateInputSize
// Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE;
ufs_res = UFS_EncryptTemplate(hScanner, TemplateInput,
                              TemplateInputSize, TemplateOutput, &TemplateOutputSize);
if (ufs_res == UFS_OK)
{
        // UFS_EncryptTemplate success
}
else
{
        // UFS_ EncryptTemplate failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_DecryptTemplate

Decryptes template.

Syntax

```
UFS_STATUS UFS_API UFS_DecryptTemplate(HUFScanner hScanner,
                                       unsigned char* pTemplateInput,
                                       int nTemplateInputSize,
                                       unsigned char* pTemplateOutput,
                                       int* pnTemplateOutputSize)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplateInput [in]: Pointer to input template data (encrypted)
- nTemplateInputSize [in]: Input template size
- pTemplateOutput [out]: Pointer to encrypted template data
- nTemplateOutputSize [in/out]: Inputs allocated size of encrypted template data; Receives output template size

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;

unsigned char   TemplateInput[MAX_TEMPLATE_SIZE];
unsigned char   TemplateOutput[MAX_TEMPLATE_SIZE];
int             TemplateInputSize;
int             TemplateOutputSize;

// After getting hScanner handle
// Get an encrypted template, TemplateInput and TemplateInputSize
// Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE;
ufs_res = UFS_DecryptTemplate(hScanner, TemplateInput,
                              TemplateInputSize, TemplateOutput, &TemplateOutputSize);
if (ufs_res == UFS_OK)
{
        // UFS_DecryptTemplate success
}
else
{
        // UFS_ DecryptTemplate failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferInfo

Gets the information of the capture image buffer.

Syntax

```
UFS_STATUS UFS_API UFS_GetCaptureImageBufferInfo(HUFScanner hScanner,
                                                 int* pnWidth,
                                                 int* pnHeight,
                                                 int* pnResolution)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pnWidth [out]: Receives the width of the capture image buffer
- pnHeight [out]: Receives the height of the capture image buffer
- pnResolution [out]: Receives the resolution of the capture image buffer

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nWidth;
int             nHeight;
int             nResolution;

// After getting hScanner handle

ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, &nWidth, &nHeight, &nResolution);
if (ufs_res == UFS_OK)
{
        // UFS_GetCaptureImageBufferInfo success
}
else
{
        // UFS_ GetCaptureImageBufferInfo failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBuffer

Copies the capture image buffer to the specified image data array.

Syntax

```
UFS_STATUS UFS_API UFS_GetCaptureImageBuffer(HUFScanner hScanner,
                                             unsigned char* pImageData)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pImageData [out]: Pointer to image data array;
  The array must be allocated bigger than the size of capture image buffer in advance.

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nWidth;
int             nHeight;
int             nResolution;
unsigned char*  pImageData;

// After getting hScanner handle

// Get capture image buffer information
ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, &nWidth, &nHeight, &nResolution);

// Allocate image buffer
pImageData = (unsigned char*)malloc(nWidth * nHeight * sizeof(unsigned char));

ufs_res = UFS_GetCaptureImageBuffer(hScanner, pImageData);
if (ufs_res == UFS_OK)
{
        // UFS_GetCaptureImageBuffer success
}
else
{
        // UFS_GetCaptureImageBuffer failure
}

// Free image buffer after using
free(pImageBuffer);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferToBMPImageBuffer

Copies the capture image buffer to the specified image data of bmp format.

Syntax

```
UFS_STATUS UFS_API UFS_GetCaptureImageBufferToBMPImageBuffer(HUFScanner hScanner,
                                                             unsigned char* pImageData,
                                                             int* pImageDataLength)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pImageData [out]: Pointer to image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance.
- pImageDataLength [out]: Pointer to bitmap image data size

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nWidth;
int             nHeight;
int             nResolution;
int             nBmpHeaderSize;
int             nBmpImageBufSize;
unsigned char*  pBmpImageBuf;

// After getting hScanner handle

// Get capture image buffer information
ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, &nWidth, &nHeight, &nResolution);

nBmpHeaderSize = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD);

// Allocate bitmap image buffer
pBmpImageBuf = (unsigned char*)malloc(nWidth * nHeight + nBmpHeaderSize);

ufs_res = UFS_GetCaptureImageBufferToBMPImageBuffer(hScanner,
                                                    pBmpImageBuf, &nBmpImageBufSize);
if (ufs_res == UFS_OK)
{
        // UFS_ GetCaptureImageBufferToBMPImageBuffer success
}
else
{
        // UFS_ GetCaptureImageBufferToBMPImageBuffer failure
}

// Free image buffer after using
free(pImageBuffer);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferTo19794_4ImageBuffer

Copies the capture image buffer to the specified image data of 19794_4 format.

Syntax

```
UFS_STATUS UFS_API UFS_GetCaptureImageBufferTo19794_4ImageBuffer(HUFScanner hScanner,
                                                                 unsigned char* pImageData,
                                                                 int* pImageDataLength,
                                                                 int nCompressType,
                                                                 float ratio)
```

Parameters

- hScanner [in]: Handle to the scanner object
- pImageData [out]: Pointer to 19794_4 format image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance.
- pImageDataLength [out]: Pointer to 19794_4 format image data size
- nCompressType [in]: Image type to copy to the specified image data of 19794_4 format

Image Type	Code	Description
UFS_19794_TYPE_RAW	0	RAW Image
UFS_19794_TYPE_WSQ	2	WSQ Image

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nWidth;
int             nHeight;
int             nResolution;
unsigned char*  pConvertedImageBuf;
int             nConvertedImageBufSize;
int             nCompressType = UFS_19794_TYPE_WSQ;
float           ratio = 0.0f;

// After getting hScanner handle

// Get capture image buffer information
ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, &nWidth, &nHeight, &nResolution);

pConvertedImageBuf = (unsigned char*)malloc(nWidth * nHeight);
ufs_res = UFS_GetCaptureImageBufferTo19794_4ImageBuffer(hScanner,
                pConvertedImageBuf, &nConvertedImageBufSize, nCompressType, ratio);
if (ufs_res == UFS_OK)
{
        // UFS_ GetCaptureImageBufferTo19794_4ImageBuffer success
}
else
{
        // UFS_ GetCaptureImageBufferTo19794_4ImageBuffer failure
}

// Free image buffer after using
free(pConvertedImageBuf);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferToWSQImageBuffer

Copies the capture image buffer to the specified image data of the WSQ format.

Syntax

```
UFS_STATUS UFS_API UFS_GetCaptureImageBufferToWSQImageBuffer(HUFScanner hScanner,
                                                             Const float ratio,
                                                             unsigned char* wsqData,
                                                             int* wsqDataLen)
```

Parameters

- hScanner [in]: Handle to the scanner object
- ratio [in]: Compression ratio of image
- wsqData [out]: Pointer to WSQ format image data; The buffer must be allocated bigger than the size of capture image buffer in advance
- wsqDataLen [out]: pointer to WSQ format image data size</fs>

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nWidth;
int             nHeight;
int             nResolution;
unsigned char*  pConvertedImageBuf;
int             nConvertedImageBufSize;
float           ratio = 0.75;

// After getting hScanner handle

// Get capture image buffer information
ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, &nWidth, &nHeight, &nResolution);

pConvertedImageBuf = (unsigned char*)malloc(nWidth * nHeight);

ufs_res = UFS_GetCaptureImageBufferToWSQImageBuffer(hScanner, nRatio,
                                pConvertedImageBuf, &nConvertedImageBufSize);
if (ufs_res == UFS_OK)
{
        // UFS_ GetCaptureImageBufferToWSQImageBuffer success
}
else
{
        // UFS_ GetCaptureImageBufferToWSQImageBuffer failure
}

// Free image buffer after using
free(pConvertedImageBuf);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferToWSQImageBufferVar

Copies the capture image buffer (cropped or expanded to the specified size) to the target image data buffer of the WSQ format.

Syntax

```
UFS_STATUS UFS_API UFS_GetCaptureImageBufferToWSQImageBufferVar(HUFScanner hScanner,
                                                                Const float ratio,
                                                                unsigned char* wsqData,
                                                                int* wsqDataLen,
                                                                int nWidth, int nHeight)
```

Parameters

- hScanner [in]: Handle to the scanner object
- ratio [in]: Compression ratio of image
- wsqData [out]: Pointer to WSQ format image data; The buffer must be allocated bigger than the size of capture image buffer in advance
- wsqDataLen [out]: pointer to WSQ format image data size
- nWidth [in]: Width to resize the capture image
- nHeight [in]: Height to resize the capture image

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nWidth;
int             nHeight;
unsigned char*  pConvertedImageBuf;
int             nConvertedImageBufSize;
float           ratio = 0.75;

// After getting hScanner handle

// Get capture image buffer information
pConvertedImageBuf = (unsigned char*)malloc(nWidth * nHeight);

ufs_res = UFS_GetCaptureImageBufferToWSQImageBufferVar(hScanner, nRatio,
                pConvertedImageBuf, &nConvertedImageBufSize, nWidth, nHeight);

if (ufs_res == UFS_OK)
{
        // UFS_ GetCaptureImageBufferToWSQImageBuffer success
}
else
{
        // UFS_ GetCaptureImageBufferToWSQImageBuffer failure
}

// Free image buffer after using
free(pConvertedImageBuf);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_DecompressWSQBMP

Initializes a UFScanner module.

Syntax

```
UFS_STATUS UFS_API UFS_DecompressWSQBMP(HUFScanner hScanner,
                                        char* wsqFile,
                                        char* bmpFile );
```

Parameters

- hScanner [in]: Handle to the scanner object
- wsqFile [in]: Specifies file name to get wsq data buffer
- bmpFile [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
char            szWsqFileName[128];
char            szBmpFileName[128]

// Get hScanner handle
// Get WSQ file name to save bmp file
// Get Bmp file from the WSQ file

ufs_res = UFS_DecompressWSQBMP(hScanner, szWsqFileName, szBmpFileName);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_DecompressWSQBMPMem

Decompress WSQ buffer and save to image data of bmp format.

Syntax

```
UFS_STATUS UFS_API UFS_DecompressWSQBMPMem(HUFScanner hScanner,
                                           unsigned char* wsqBuffer,
                                           int wsqBufferLen,
                                           unsigned char* bmpBuffer,
                                           int* bmpBufferLen );
```

Parameters

- hScanner [in]: Handle to the scanner object
- wsqBuffer [in]: Pointer to WSQ format image data
- wsqBufferLen [in]: Size of WSQ format image data
- bmpBuffer [out]: Pointer to bmp image data; The array must be allocated bigger than the size of capture image buffer in advance.
- bmpBufferLen [out]: pointer to bmp image data size

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
int             nWidth;
int             nHeight;
int             nResolution;
int             nBmpHeaderSize;
unsigned char*  pWsqBuffer;
int             nWsqBufferLen;
unsigned char*  pBmpBuffer;
int             nBmpBufferLen

// Get hScanner handle
// Get WSQ data (ex.UFS_GetCaptureImageBufferToWSQImageBuffer)

ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, &nWidth, &nHeight, &nResolution);

nBmpHeaderSize = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + sizeof(RGBQUAD)

pBmpBuffer = (unsigned char*)malloc(nWidth * nHeight * nBmpHeaderSize);

ufs_res = UFS_DecompressWSQBMPMem(hScanner, pWsqBuffer, nWsqBufferLen, pBmpBuffer,
                                                                    nBmpBufferLen);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
free(pImageBuffer);
free(pWsqBuffer);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_DrawCaptureImageBuffer

Draws the fingerprint image which is acquired using UFS_CaptureSingleImage or UFS_StartCapturing. This function is not supported on java.

Syntax

```
UFS_STATUS UFS_API UFS_DrawCaptureImageBuffer(HUFScanner hScanner,
                                              HDC hDC,
                                              int nLeft,
                                              int nTop,
                                              int nRight,
                                              int nBottom,
                                              int bCore );
```

Parameters

- hScanner [in]: Handle to the scanner object
- hDC [in]: Handle to the DC where the fingerprint image is drawn
- nLeft [in]: Specifies the logical x-coordinate of the upper-left corner of the rectangle
- nTop [in]: Specifies the logical y-coordinate of the upper-left corner of the rectangle
- nRight [in]: Specifies the logical x-coordinate of the lower-right corner of the rectangle
- nBottom [in]: Specifies the logical y-coordinate of the lower-right corner of the rectangle
- bCore [in]: Specifies whether the core of fingerprint is drawn of not

Return value

UFS_STATUS

Remarks

Examples

```
// Get hScanner handle

UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
HDC             hDC;
int             nLeft;
int             nTop;
int             nRight;
int             nBottom;
int             bCore;

// Get HDC and determine rectangle to draw image, hDC, nLeft, nTop, nRight, nBottom
// Determine core to be drawn, bCore

ufs_res = UFS_DrawCaptureImageBuffer(hScanner, hDC, nLeft, nTop, nRight, nBottom, bCore);
if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_STATUS

UFS_DrawFeatureBuffer

Draws the fingerprint image which is acquired using UFS_CaptureSingleImage or UFS_StartCapturing.
This function is not supported on java. And should be called after extraction from last captured fingerprint image. If extraction is not performed from the last captured image, this function will not draw the feature in image frame.

Syntax

```
UFS_STATUS UFS_API UFS_DrawFeatureBuffer(HUFScanner hScanner,
                                         HDC hDC,
                                         int nLeft,
                                         int nTop,
                                         int nRight,
                                         int nBottom,
                                         int bCore );
```

Parameters

- hScanner [in]: Handle to the scanner object
- hDC [in]: Handle to the DC where the fingerprint image is drawn
- nLeft [in]: Specifies the logical x-coordinate of the upper-left corner of the rectangle
- nTop [in]: Specifies the logical y-coordinate of the upper-left corner of the rectangle
- nRight [in]: Specifies the logical x-coordinate of the lower-right corner of the rectangle
- nBottom [in]: Specifies the logical y-coordinate of the lower-right corner of the rectangle
- bCore [in]: Specifies whether the core of fingerprint is drawn of not

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
HDC             hDC;
int             nLeft;
int             nTop;
int             nRight;
int             nBottom;
int             bCore;

// Get hScanner handle
// Get HDC and determine rectangle to draw image, hDC, nLeft, nTop, nRight, nBottom
// Determine core to be drawn, bCore

ufs_res = UFS_DrawFeatureBuffer(hScanner, hDC, nLeft, nTop, nRight, nBottom, bCore);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_STATUS

UFS_SaveCaptureImageBufferToBMP

Saves the capture image buffer to the specified file of the bitmap format.

Syntax

```
UFS_STATUS UFS_API UFS_SaveCaptureImageBufferToBMP(HUFScanner hScanner,
                                                   char* szFileName );
```

Parameters

- hScanner [in]: Handle to the scanner object
- szFileName [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
char            szFileName[128];

// Get hScanner handle
// Get file name, szFileNam

ufs_res = UFS_SaveCaptureImageBufferToBMP(hScanner, szFileName);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SaveCaptureImageBufferTo19794_4

Saves the capture image buffer to the specified file of the 19794_4 format.

Syntax

```
UFS_STATUS UFS_API UFS_SaveCaptureImageBufferTo19794_4(HUFScanner hScanner,
                                                       char* szFileName,
                                                       int nCompressType,
                                                       float ratio);
```

Parameters

- hScanner [in]: Handle to the scanner object
- szFileName [in]: Specifies file name to save image buffer
- nCompressType [in]: Image type to copy to the specified image data of 19794_4 format

Image Type	Code	Description
UFS_19794_TYPE_RAW	0	Raw Image
UFS_19794_TYPE_WSQ	2	Wsq Image

- ratio [in]: Compression ratio of image

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS        ufs_res;
HUFScanner        hScanner = NULL;
char              szFileName[128];
int               nCompressType = UFS_19794_TYPE_WSQ;
float             ratio = 0.0f;

// Get hScanner handle
// Get file name, szFileNam

ufs_res = UFS_SaveCaptureImageBufferTo19794_4(hScanner, szFileName, nCompressType, ratio);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SaveCaptureImageBufferToWSQ

Saves the capture image buffer to the specified file of the WSQ format.

Syntax

```
UFS_STATUS UFS_API UFS_SaveCaptureImageBufferToWSQ(HUFScanner hScanner,
                                                   const float ratio,
                                                   char* szFileName );
```

Parameters

- hScanner [in]: Handle to the scanner object
- ratio [in]: Compression ratio of image
- szFileName [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

```
ufs_res = UFS_SaveCaptureImag UFS_STATUSufs_res;
HUFScanner        hScanner = NULL;
char              szFileName[128];

// Get hScanner handle
// Get file name; szFileName

eBufferToWSQ(hScanner, ratio, szFileName);
if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SaveCaptureImageBufferToWSQVar

Saves the capture image buffer (cropped or expanded to the specified size) to the target file of the WSQ format.

Syntax

```
UFS_STATUS UFS_API UFS_SaveCaptureImageBufferToWSQVar(HUFScanner hScanner,
                                                      const float ratio,
                                                      char* szFileName,
                                                      int nWidth,
                                                      int nHeight );
```

Parameters

- hScanner [in]: Handle to the scanner object
- ratio [in]: Compression ratio of image
- szFileName [in]: Specifies file name to save image buffer
- nWidth [in]: Width to resize the capture image
- nHeight [in]: Height to resize the capture image

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      ufs_res;
HUFScanner      hScanner = NULL;
char            szFileName[128];
int             nWidth;
int             nHeight;

// Get hScanner handle
// Get file name; szFileName
// Get size of capture image to resize; nWidth,nHeight

ufs_res = UFS_SaveCaptureImageBufferToWSQVar(hScanner, ratio, szFileName, nWidth, nHeight);
if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_ClearCaptureImageBuffer

Clears the capture image buffer stored to the internal buffer.

Syntax

```
UFS_STATUS UFS_API UFS_ClearCaptureImageBuffer(HUFScanner hScanner );
```

Parameters

- hScanner [in]: Handle to the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
HUFScanner      hScanner = NULL;

// Get hScanner handle

ufs_res = UFS_ClearCaptureImageBuffer(hScanner);
if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetErrorString

Gets the error string for specified UFS_STATUS value.

Syntax

```
UFS_STATUS UFS_API UFS_GetErrorString(UFS_STATUS res, char* szErrorString );
```

Parameters

- res [in]: Status return value
- szErrorString [out]: Receives error string

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
char            strError[128];

// Get status return code, ufs_res

ufs_res = UFS_GetErrorString(ufs_res, strError);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetTemplateType

Gets the template type value.

Syntax

```
UFS_STATUS UFS_API UFS_GetTemplateType(HUFScanner hScanner,
                                       int* nTemplateType );
```

Parameters

- hScanner [in]: Handle to the scanner object
- nTemplateType [out]: Receives the parameter value of specified parameter type; 'pValue' must point to adequate type that is matched with the parameter type

Template type	Code	Description
UFS_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFS_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFS_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
HUFScanner      hScanner = NULL;
int             nTemplateType;

// Get hScanner handle

ufs_res = UFS_GetTemplateType(hScanner, &nTemplateType);
if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SetTemplateType

Sets the template type value.

Syntax

```
UFS_STATUS UFS_API UFS_SetTemplateType(HUFScanner hScanner,
                                       int nTemplateType );
```

Parameters

- hScanner [in]: Handle to the scanner object
- nTemplateType [in]: Parameter type; one of template type

Template type	Code	Description
UFS_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFS_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFS_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
HUFScanner      hScanner = NULL;
int             nTemplateType;

// Get hScanner handle

nTemplateType = UFS_TEMPLATE_TYPE_SUPREMA;

ufs_res = UFS_SetTemplateType(hScanner, nTemplateType);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SelectTemplate

Selects n number of good templates from m number of input templates.

Syntax

```
UFS_STATUS UFS_API UFS_SelectTemplate(HUFScanner hScanner,
                                      unsigned char** ppTemplateInput,
                                      int* pnTemplateInputSize,
                                      int nTemplateInputNum,
                                      unsigned char** ppTemplateOutput,
                                      int* pnTemplateOutputSize,
                                      int nTemplateOutputNum );
```

Parameters

- hScanner [in]: Handle to the scanner object
- ppTemplateInput [in]: Array pointer to the input template arrays
- pnTemplateInputSize [in]: Array pointer to input templates' size
- nTemplateInputNum [in]: Number of input templates
- ppTemplateOutput [out]: Array pointer to the output template arrays
- pnTemplateOutputSize [out]: Array pointer to the output templates' size
- nTemplateOutputNum [in]: Number of output templates;
  should be less than input template number by more than one

Return value

UFS_STATUS

Remarks

Examples

```
unsigned char          Template[MAX_TEMPLATE_SIZE];
int                    TemplateSize;
int                    nEnrollQuality;
UFS_STATUS             ufs_res;
int                    i = ; // sample number
int                    inputNum = 4;
int                    outputNum = 2;

While(1) {
        // capture a fingerprint image

        ufs_res = UFS_Extract(hScanner, Template, &TemplateSize, &nEnrollQuality);

        // if UFS_Extract is succeed
        memcpy(InputTemplateArray[i], Template, TemplateSize);
        InputTemplateSizeArray[i] = TemplateSize;

        i++;

        if (i == inputNum)

                break;

}

ufs_res = UFS_SelectTemplate(hScanner, InputTemplateArray, InputTemplateSizeArray, inputNum,
                                        OutputTemplateArray, OutputTemplateSizeArray, outputNum);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SelectTemplateEx

Selects n number of good templates from m number of input templates.
This is extended version of UFS_SelectTemplate function to accommodate large size template.

Syntax

```
UFS_STATUS UFS_API UFS_SelectTemplateEx(HUFScanner hScanner,
                                        int nBufferSize,
                                        unsigned char** ppTemplateInput,
                                        int* pnTemplateInputSize,
                                        int nTemplateInputNum,
                                        unsigned char** ppTemplateOutput,
                                        int* pnTemplateOutputSize,
                                        int nTemplateOutputNum);
```

Parameters

- hScanner [in]: Handle to the scanner object
- nBufferSize [in]: Template buffer size
- ppTemplateInput [in]: Array pointer to the input template arrays
- pnTemplateInputSize [in]: Array pointer to the input templates' size
- nTemplateInputNum [in]: Number of input templates
- ppTemplateOutput [out]: Array pointer to the output template arrays
- pnTemplateOutputSize [out]: Array pointer to the output templates' size
- nTemplateOutputNum [in]: Number of output templates; should be less than input template number by more than one

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS        usf_res;
unsigned char     Template[MAX_TEMPLATE_SIZE];
int               TemplateSize;
int               nEnrollQuality
int               i = ; // sample number
Int               inputNum = 4;
int               outputNum = 2;

While(1)
{
        // capture a fingerprint image

        ufs_res = UFS_Extract(hScanner, Template, &TemplateSize, &nEnrollQuality);

        // if UFS_Extract is succeed
        memcpy(InputTemplateArray[i], Template, TemplateSize);
        InputTemplateSizeArray[i] = TemplateSize;
        i++;
        if (i == inputNum)
                break;
}
ufs_res = UFS_SelectTemplateEx(hScanner, MAX_TEMPLATE_SIZE, InputTemplateArray,
                                InputTemplateSizeArray, inputNum, OutputTemplateArray,
                                OutputTemplateSizeArray, outputNum);

// UFS_SelectTemplate is succeed
if (ufs_res == UFS_OK)
{
        //UFS_Init success
        // If you want to enroll the output templates, move OutputTemplateArray and
           OutputTemplateSizeArray data to your template array for enrollment or database.
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetFPQuality

Calculates the quality score of a fingerprint according to UFS_PARAM_FPQUALITY_MODE parameter.

NFIQ 2.0 (NIST Fingerprint Image Quality) is the quality score of an image as defined in the draft.
To use NFIQ 2.0, you need to set the UFS_SetParameter API as follows.
1) Set UFS_PARAM_FPQUALITY_MODE to UFS_NQS_MODE_NFIQ2.
2) Set the value of UFS_PARAM_NFIQ2_FILE to the path where the NFIQ 2.0 Library file is located. (For example, the NFIQ2.0 Library file in the BioMini PC SDK package is NFIQ2.dll.)

Syntax

```
UFS_STATUS UFS_API UFS_GetFPQuality(HUFScanner hScanner,
                                    unsigned char* pFPImage,
                                    int nWidth,
                                    int nHeight,
                                    int* pnFPQuality );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pFPImage [in]: Raw capture image data
- nWidth [in]: Width of capture image data
- nHeight [in]: Height of capture image data
- pnFPQuality [out]: Quality score of image data

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS       usf_res;
HUFScanner        hScanner = NULL;
unsigned char    *pCaptImageBuf;
int              nCaptImageWidth;
int              nCaptImageHeight;
int              nFPQuality;

// Capture a fingerprint image
// Get Capture Image Data; pCaptImageBuf, nCaptImageWidth, nCaptImageHeigh

ufs_res = UFS_GetFPQuality(hScanner, pCaptImageBuf, nCaptImageWidth,
                           nCaptImageWidth, &nFPQuality);
if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetFeatureNumber

Get the number of Minutiae from the template data

Syntax

```
UFS_STATUS UFS_API UFS_GetFeatureNumber (HUFScanner hScanner,
                                         unsigned char* pTemplate,
                                         int nTemplateSize,
                                         int* pnFeatureNum );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplate [in]: Template data
- nTemplateSize [in]: Size of template data
- pnFeatureNum [out]: The number of minutiae from pTemplate

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS       usf_res;
HUFScanner        hScanner = NULL;
unsigned char    Template[MAX_TEMPLATE_SIZE];
int              TemplateSize;
int              nFeatureNum;
int              nEnrollQuality;

// Capture a fingerprint image

ufs_res = UFS_Extract(hScanner, Template, &TemplateSize, &nEnrollQuality);

ufs_res = UFS_GetFeatureNumber(hScanner, Template, TemplateSize, &nFeatureNum);

if (ufs_res == UFS_OK)
{
        //UFS_Init success
}
else
{
        //UFS_Init failure
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_EnrollUI

Generate the fingerprint enrollment dialog. This function can be called after executing UFS_Init. Enrolling a fingerprint is extracting a template from finger and saving the template. Below sample's UFS_EnrollUI function captures a fingerprint image after setting the template type. And extracts a template from captured fingerprint image. The extracted template will be saved in a specific template array, which is a parameter of the UFS_EnrollUI function. It supported only for Windows environment.

*Constraints
– You should have 'img' folder to use graphical backgrounds and buttons. The application uses the img folder should be at the upper level folder. For example, if the application is at the /bin/sample, 'img' folder should be at the location of /bin/sample/img. – Enrollment UI is based on COM interface. Thus you should register dll file before use. You can use the pre-coded script (register_enrollui.bat) to register the dll file, or simple type the command 'regsvr32.exe IEnrollUI.dll' at the command prompt.

Syntax

```
UFS_STATUS UFS_API UFS_EnrollUI(HUFScanner hScanner,
                                int nTimeout,
                                int nOptions, BYTE* pUF_FIR_Buf,
                                int* pUF_FIR_Buf_Len, BYTE* pISO_FIR_Buf,
                                int* pISO_FIR_Buf_Len,
                                char* pImages_Path,    BYTE* pImage_Buf = NULL,
                                int* pImage_Buf_Len= NULL)
```

Parameters

- hScanner [in]: Handle to the scanner object
- nTimeout [in]: Timeout of the capture
- nOptions [in]: Options for enrollment. Matching level, image Quality, number of fingerprints for enrollment, number of templates per finger
- pUF_FIR_Buf [out]: Pointer to the byte array for suprema template. This data pointer is assigned by maximum 1024*20
- pUF_FIR_Buf_Len [out]: Pointer to the int array for length of suprema template buffer
- pISO_FIR_Buf [out]: Pointer to the byte array for ISO template. This data pointer is assigned by maximum 1024 * 20
- pISO_FIR_Buf_Len [out]: Pointer to the int array for length of ISO template buffer
- pImages_Path [in]: Path to captured images to be saved. If NULL value is passed, nothing will be saved
- pImage_Buf [out]: Pointer to the byte array for image buffer. This data pointer is assigned by maximum 320 * 480
- pImage_Buf_Len [out]: Pointer to the int array for length of image buffer

Return value

UFS_STATUS

Remarks

Examples

```
// (1) initialize the buffer
BYTE* pUFBuf = new BYTE[1024*20];
memset(pUFBuf, , 1024*20);
int* pUFBufSize = new int[20];
memset(pUFBufSize, , 20);

BYTE* pISOBuf = new BYTE[1024*20];
memset(pISOBuf, , 1024*20);
int* pISOBufSize = new int[20];
memset(pISOBufSize, , 20);

int nFingersToEnroll = 10;
int nTemplatesPerFinger = 2;

// (2) set the options
int nOptions = UF_PACK_SECURITY(m_nSecurityLevel+1) |
                    UF_PACK_QUALITY(m_quality) |
                    UF_PACK_NFINGERS(nFingersToEnroll) |
                    UF_PACK_NTEMPLATES(nTemplatesPerFinger);

// (3) execute the enrollment ui api
UFS_EnrollUI(hScanner, m_nTimeout, nOptions, pUFBuf,
                  pUFBufSize, pISOBuf, pISOBufSize, "c:\\", null, null);

// (4) get the templates buffer
memcpy(m_pTemplateBuf, pUFBuf, 1024*20*sizeof(BYTE));
memcpy(m_pTemplateBufSize, pUFBufSize, 20*sizeof(int));

// (5) release buffer
delete[] pUFBuf;
delete[] pUFBufSize;
delete[] pISOBuf;
delete[] pISOBufSize
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_VerifyUI

Generate the fingerprint verification dialog. This function can be called after executing UFS_Init and UFS_EnrollUI. Two fingerprints can be verified whether they are matched or not. Below sample's UFS_VerifyUI function captures a fingerprint image and extracts a template from the image. And execute 1:1 matching using extracted template and templates enrolled from UFS_EnrollUI.

*Constraints
– Refer to the UFS_EnrollUI constraints.

Syntax

```
UFS_STATUS UFS_API UFS_VerifyUI(HUFScanner hScanner,
                                int nTimeout,
                                int nOptions,
                                int nFPTemplateType, BYTE* pFIR_BUF,
                                int* pFIR_Buf_Len,
                                char* pImages_Path,
                                int* nFingerIndex);
```

Parameters

- hScanner [in]: Handle to the scanner object
- nTimeout [in]: Timeout of the capture
- nOptions [in]: Options for enrollment. Matching level, image Quality, number of fingerprints for enrollment, number of templates per finger
- nFPTemplateType [in]: Template type for matching enrolled templates with captured fingerprint
- pFIR_Buf [out]: Pointer to the byte array for template
- pFIR_Buf_Len [out]: Pointer to the int array for length of template buffer
- pImages_Path [in]: Path to captured images
- nFingerIndex [in]: Matched finger index from enrolled templates. If this value is -1, the matching result is failed

Return value

UFS_STATUS

Remarks

Examples

```
int nFingersToEnroll = 10;
int nTemplatesPerFinger = 2;

// (1) set the options
int nOptions = UF_PACK_SECURITY(m_nSecurityLevel+1) |
                    UF_PACK_QUALITY(m_quality) |
                    UF_PACK_NFINGERS(nFingersToEnroll) |
                    UF_PACK_NTEMPLATES(nTemplatesPerFinger)
// (2) excute the enrollment ui ap
UFS_VerifyUI(hScanner, m_nTimeout, UFS_TEMPLATE_TYPE_SUPREMA, nOptions,
                                        pBuf, pBufSize, "verfication.bmp", &nFingerIndex);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_CaptureSingleUI

Performs same as UFS_CaptureSingle and Popup Window appears once the capturing starts to show a captured image then disappears.

Syntax

```
UFS_STATUS UFS_API UFS_CaptureSingleUI(HUFScanner hScanner,
                                       int nTimeout,
                                       int nOptions, BYTE* pUFImageBuf,
                                       int* pUFImageWidth,
                                       int* pUFImageHeight,
                                       char* pImages_Path,
                                       int* nFPQuality);
```

Parameters

- hScanner [in]: Effective handle for connected BioMini
- nTimeout [in]: Applicable timeout parameter for capture single function
- nOptions [in]: Same as UFS_EnrollUI option
- pUFImageBuf [in]: Buffer of a captured image (The memory buffer has to me managed by the user)
- pUFImageWidth, int* pUFImageHeight [in]: Width / Height of a captured image (pixel)
- pImages_Path [in]: Path to save an image
- nFPQuality [in]: Returns same score as UFS_GetFPQuality

Return value

UFS_STATUS

Remarks

Examples

```
UFS_STATUS ufs_res = UFS_GetCaptureImageBufferInfo(hScanner ,&nWidth ,
                                                    &nHeight, &nResolution );

unsigned char *pbImage = (unsigned char*)malloc(nWidth * nHeight);

ZeroMemory(pbImage ,    , nWidth* nHeight);

int nOptions =
            UF_PACK_SECURITY(hWnd->m_nSecurityLevel+1) |
            UF_PACK_QUALITY(hWnd->m_quality);

ufs_res = UFS_CaptureSingleUI(hScanner, hWnd->m_nTimeout,
                                nOptions, pbImage, &nWidth , &nHeight, NULL, &nQuality);

hWnd->AddMessage( "UFS_CaptureSingleUI FPQuality %s : %d\r\n",
ufs_res==UFS_OK?"OK":"Failed", nQuality);

free(pbImage);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCompanyID

Get a company ID assigned to the scanner

Syntax

```
UFS_STATUS UFS_API UFS_GetCompanyID(HUFScanner hScanner,
                                    char* szCompanyID)
```

Parameters

- hScanner [in]: Effective handle for connected BioMini
- szCompanyID [out]: Company ID buffer

Return value

UFS_STATUS

Remarks

Examples

```
char cid[3] = {, };

if( UFS_GetCompanyID(hScanner, cid) == UFS_OK ) {
        //success.
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_STATUS

Every function in a UFScanner module returns UFS_OK when it succeeds. When it fails, it returns a value corresponding to an error code. Please find the error code on the followings if you'd like to know the information about the UFS_STATUS (integer) value.

Code	Value	Description
UFS_OK	0	Success
UFS_ERROR	-1	General error
UFS_ERR_DEVICE_NOT_RESPOND	-10	Device is not responded
UFS_ERR_CAPTURE_TIMEOUT	-11	The time for capturing exceeds
UFS_ERR_USB_TIMEOUT	-12	The time for USB transfer exceeds
UFS_ERR_NO_LICENSE	-101	Device is not connected or License is not located
UFS_ERR_LICENSE_NOT_MATCH	-102	License does not match
UFS_ERR_LICENSE_EXPIRED	-103	License has expired
UFS_ERR_NOT_SUPPORTED	-111	This function is not supported
UFS_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFS_ERR_UNKNOWN_TEMPLATE_FORMAT	-120	Abnormal template recognized
UFS_ERR_INVALIDE_ENCRYPTION	-121	Template buffer, encrypted template size is incorrect
UFS_ERR_INVALID_MODE	-122	Device is sleep mode
UFS_ERR_ALREADY_INITIALIZED	-201	Module is already initialized
UFS_ERR_NOT_INITIALIZED	-202	Module is not initialized
UFS_ERR_DEVICE_NUMBER_EXCEED	-203	Device number exceeds
UFS_ERR_LOAD_SCANNER_LIBRARY	-204	Error on loading the library of a scanner
UFS_ERR_CAPTURE_RUNNING	-211	Capturing is started using UFS_CaptureSingleImage of UFS_StartCapturing
UFS_ERR_CAPTURE_FAILED	-212	Capturing is timeout or aborted
UFS_ERR_FAKE_FINGER	-221	Fake finger is detected
UFS_ERR_FINGER_ON_SENSOR	-231	Remove finger from sensor
UFS_ERR_TIMEOUT	-241	Time out
UFS_ERR_NOT_GOOD_IMAGE	-301	Input image is not good
UFS_ERR_EXTRACTION_FAILED	-302	Extraction is failed
UFS_ERR_ID_NOT_FOUND	-500	ID not found on device
UFS_ERR_NO_MATCH_FOUND	-501	Not registered fingerprint
UFS_ERR_NOT_IDENTICAL	-502	Not matched
UFS_ERR_CANNOT_ENROLL	-510	Cannot enroll fingerprint
UFS_ERR_NO_DATA	-511	No data found on device
UFS_ERR_NOT_AUTHORIZED	-520	Not authorized to use this API

Code	Value	Description
UFS_ERR_CORE_NOT_DETECTED	-351	Core is not detected
UFS_ERR_CORE_TO_LEFT	-352	Move finger to left
UFS_ERR_CORE_TO_LEFT_TOP	-353	Move finger to left-top
UFS_ERR_CORE_TO_TOP	-354	Move finger to top
UFS_ERR_CORE_TO_RIGHT_TOP	-355	Move finger to right-top
UFS_ERR_CORE_TO_RIGHT	-356	Move finger to right
UFS_ERR_CORE_TO_RIGHT_BOTTOM	-357	Move finger to right-bottom
UFS_ERR_CORE_TO_BOTTOM	-358	Move finger to bottom
UFS_ERR_CORE_TO_LEFT_BOTTOM	-359	Move finger to left-bottom
UFS_ERR_FINGER_TOO_RIGHT	-401	Finger is too close to the right edge
UFS_ERR_FINGER_TOO_LEFT	-402	Finger is too close to the left edge
UFS_ERR_FINGER_TOO_TOP	-403	Finger is too close to the top
UFS_ERR_FINGER_TOO_BOTTOM	-404	Finger is too close to the bottom
UFS_ERR_FINGER_TIP	-405	Do not put the fingertip

In case UFS_PARAM_DETECT_CORE is “1” (=use), you may get the following error

UFMatcher Functions

UFM_Create

Creates a matcher object.

Syntax

```
UFM_STATUS UFM_API UFM_Create(HUFMatcher* phMatcher);
```

Parameters

- phMatcher [out]: Pointer to handle of the matcher object

Return value

UFM_STATUS

Remarks

Examples

```
UFM_STATUS     ufm_res;
HUFMatcher      hMatcher = NULL;

ufm_res = UFM_Create(&hMatcher);
if (ufm_res == UFM_OK)
{
        //UFM_Create success
}
else
{
        //UFM_Create failure
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_Delete

Deletes specified matcher object.

Syntax

```
UFM_STATUS UFM_API UFM_Delete(HUFMatcher pMatcher);
```

0

Parameters

- pMatcher [in]: Handle to the matcher object

Return value

UFM_STATUS

Remarks

Examples

```
UFM_STATUS     ufm_res;
HUFMatcher      hMatcher = NULL;

//Create hMatcher and use

ufm_res = UFM_Delete(hMatcher);

if (ufm_res == UFM_OK)
{
        //UFM_Delete success
}
else
{
        //UFM_Delete failure
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_GetParameter

Gets parameter value of UFMatcher module

Syntax

```
UFM_STATUS UFM_API UFM_GetParameter(HUFMatcher pMatcher,
                                    int nParam,
                                    void* pValue );
```

Parameters

- pMatcher [in]: Handle to the matcher object
- nParam [in]: Parameter type; one of parameters

Parameter	Code	Description	Default value
UFM_PARAM_FAST_MODE	301	Fast Mode (0: not use fast mode, 1: use fast mode)	1
UFM_PARAM_SECURITY_LEVEL	302	Set the False Accept Ratio(FAR) (1: Below 1%, 2: Below 0.1%, 3: Below 0.01%, 4: Below 0.001%, 5: Below 0.0001%, 6: Below 0.00001%, 7: Below 0.000001%)	4
UFM_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFM_PARAM_AUTO_ROTATE	321	Rotate Mode(0: not use rotate mode, 1: use rotate mode)	0
UFM_PARAM_SDK_VERSION	210	SDK Version (get only)	-
UFM_PARAM_SDK_COPYRIGHT	211	SDK Copyright (get only)	

- pValue [out]: Receives parameter value of specified parameter type;
  pValue must point to adequate storage type matched to parameter type

Return value

UFM_STATUS

Remarks

Examples

```
UFM_STATUS     ufm_res;
HUFMatcher     hMatcher = NULL;
int            nValue;

//Create hMatcher

//Get fast mode
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_FAST_MODE, &nValue)

//Get security level
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL, &nValue);

//Get use SIF
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_USE_SIF, &nValue);
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_SetParameter

Sets parameter value of UFMatcher module.

Syntax

```
UFM_STATUS UFM_API UFM_SetParameter(HUFMatcher pMatcher,
                                   int nParam,
                                   void* pValue );
```

Parameters

- pMatcher [in]: Handle to the matcher object
- nParam [in]: Parameter type; one of parameters

Parameter	Code	Description	Default value
UFM_PARAM_FAST_MODE	301	Fast Mode (0: not use fast mode, 1: use fast mode)	1
UFM_PARAM_SECURITY_LEVEL	302	Set the False Accept Ratio(FAR) (1: Below 1%, 2: Below 0.1%, 3: Below 0.01%, 4: Below 0.001%, 5: Below 0.0001%, 6: Below 0.00001%, 7: Below 0.000001%)	4
UFM_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFM_PARAM_AUTO_ROTATE	321	Rotate Mode(0: not use rotate mode, 1: use rotate mode)	0

Return value

UFM_STATUS

Remarks

Examples

```
UFM_STATUS    ufm_res;
HUFMatcher    hMatcher = NULL;
int           nValue;

// Create hMatcher

// Set fast mode to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_FAST_MODE, &nValue);

// Set security level to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL, &nValue); }

// Set use SIF to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_USE_SIF, &nValue);
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_Verify

Compares two extracted templates.

Syntax

```
UFM_STATUS UFM_API UFM_Verify(HUFMatcher pMatcher,
                              unsigned char* pTemplate1,
                              int nTemplate1Size,
                              unsigned char* pTemplate2,
                              int nTemplate2Size,
                              int* bVerifySucceed);
```

Parameters

- pMatcher [in]: Handle to the matcher object
- pTemplate1 [in]: Pointer to the template1
- nTemplate1Size [in]: Specifies the size of the template1
- pTemplate2 [in]: Pointer to the template2
- nTemplate2Size [in]: Specifies the size of the template2
- bVerifySucceed [out]: Receives, whether verification is succeed;
  1: verification is succeed, 0: verification is failed

Return value

UFM_STATUS

Remarks

Examples

```
// Assume template size is 1024 bytes
#define MAX_TEMPLATE_SIZE 1024

UFM_STATUS     ufm_res;
HUFMatcher     hMatcher = NULL;
unsigned char  Template1[MAX_TEMPLATE_SIZE];
unsigned char  Template2[MAX_TEMPLATE_SIZE];
int            nTemplate1Size;
int            nTemplate2Size;
int            bVerifySucceed;

//Create hMatcher
//Get two templates, Template1 and Template2

ufm_res = UFM_Verify(hMatcher, Template1, nTemplate1Size, Template2, nTemplate2Size,
                                                                    &bVerifySucceed);
if ((ufm_res == UFM_OK)
{
        //UFM_Verify success
        if (bVerifySucceed)
        {
                //Template1 is matched to Template2
        }
        else
        {
                // Template1 is not matched to Template2
        }
}
else
{
        //UFM_Verify failure
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_VerifyEx

Performs same as UFM_Verify, and returns matching score by 6th parameter
(matching score in between 0~1, idle match as the score is close to 1)

Syntax

```
UFM_STATUS UFM_API UFM_VerifyEx(HUFMatcher hMatcher,
                                unsigned char* pTemplate1,
                                int nTemplate1Size,
                                unsigned char* pTemplate2,
                                int nTemplate2Size,
                                float* fScore,
                                int* bVerifySucceed);
```

Parameters

- hMatcher [in]: Handle to the matcher object
- pTemplate1 [in]: Pointer to the template1
- nTemplate1Size [in]: Specifies the size of the template1
- pTemplate2 [in]: Pointer to the template2
- nTemplate2Size [in]: Specifies the size of the template2
- fScore [out]: Matching score between pTemplate1 and pTemplate2
- bVerifySucceed[out]: Receives, whether verification is succeed;
  1: verification is succeed, 0: verification is failed

Return value

UFM_STATUS

Remarks

Examples

```
//Assume template size is 1024 bytes
#define MAX_TEMPLATE_SIZE 1024

UFM_STATUS     ufm_res;
HUFMatcher     hMatcher = NULL;
unsigned char  Template1[MAX_TEMPLATE_SIZE];
unsigned char  Template2[MAX_TEMPLATE_SIZE];
int            nTemplate1Size;
int            nTemplate2Size;
float          fScore;
int            bVerifySucceed;

//Create hMatcher
//Get two templates, Template1 and Template2

ufm_res = UFM_Verify(hMatcher, Template1, nTemplate1Size, Template2,
                                nTemplate2Size, &fScore, &bVerifySucceed);

if ((ufm_res == UFM_OK)
{
        //UFM_Verify success
        if (bVerifySucceed)
        {
                //Template1 is matched to Template2
        }
        else
        {
                //Template1 is not matched to Template2
        }
}
else
{
        //UFM_Verify failure
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_Identify, UFM_IdentifyMT

Compares a template with given template array. UFM_IdentifyMT function uses multi threads internally for faster identifying in multi-core systems.

Syntax

```
UFM_STATUS UFM_API UFM_Identify(HUFMatcher pMatcher,
                                unsigned char* pTemplate1,
                                int nTemplate1Size,
                                unsigned char** ppTemplate2,
                                int* pnTemplate2Size,
                                int nTemplate2Num,
                                int nTimeout,
                                int* pnMatchTemplate2Index);
UFM_STATUS UFM_API UFM_IdentifyMT(HUFMatcher pMatcher,
                                  unsigned char* pTemplate1,
                                  int nTemplate1Size,
                                  unsigned char** ppTemplate2,
                                  int* pnTemplate2Size,
                                  int nTemplate2Num,
                                  int nTimeout,
                                  int* pnMatchTemplate2Index);
```

Parameters

- pMatcher [in]: Handle of the matcher object
- pTemplate1 [in]: Pointer to the template
- nTemplate1Size [in]: Specifies the size of the template
- ppTemplate2 [in]: Pointer to the template array
- pnTemplate2Size [in]: Pointer to the template size array
- nTemplate2Num [in]: Specifies the number of templates in the template array
- nTimeout [in]: Specifies maximum time for identifying in milliseconds;
  If elapsed time for identifying exceeds nTimeout, function stops further identifying and returns UFM_ERR_MATCH_TIMEOUT; 0 means infinity
- pnMatchTemplate2Index [out]: Receives the index of matched template in the template array;
  -1 means pTemplate1 is not matched to all of templates in ppTemplate2

Return value
UFM_STATUS

Remarks

Examples

```
//Assume template size is 1024 bytes
#define MAX_TEMPLATE_SIZE 1024 UFM_STATUS ufm_res;

HUFMatcher hMatcher = NULL;
unsigned char Template1[MAX_TEMPLATE_SIZE];
unsigned char** ppTemplate2;
int nTemplate1Size;
int* pnTemplate2Size;
int nTemplate2Num;
int nTimeout;
int nMatchTemplate2Index;
int i;

//Create hMatcher

//Get input template from user, Template

//Make template array from DB or something
//Get number of template to nTemplate2Num
ppTemplate2 = (unsigned char**)malloc(nTemplate2Num * sizeof(unsigned char*));

pnTemplate2Size = (int*)malloc(nTemplate2Num * sizeof(int));

for (i = ; i < nTemplate2Num; i++)
{
        ppTemplate2[i] = (unsigned char*)malloc(MAX_TEMPLATE_SIZE * sizeof(unsigned char));


        //Copy i th template to ppTemplate2[i]

        //Set i th template size to pnTemplateSize[i]

}
```

```
//Set match timeout to nTimeout
ufm_res = UFM_Identify(hMatcher, Template1, Template1Size, ppTemplate2,
                                pnTemplate2Size, nTemplate2Num, nTimeout, &nMatchTemplate2Index);

if (ufm_res == UFM_OK)
{
        //UFS_Init success
        if (nMatchTemplate2Index != -1)
        {
                //Input fingerprint Template1 is matched to
                 ppTemplate2[nMatchTemplate2Index]
        }
        else(ufm_res == UFM_OK)
        {
                // Input fingerprint is not in ppTemplate2
        }
}
else
{
        //UFM_Identify is failed
        //Use UFM_GetErrorString function to show error string
}
//Free template array
free(pnTemplate2Size);
for (i = ; i < nTemplate2Num; i++)
{
        free(ppTemplate2[i]);
}
free(ppTemplate2);
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_AbortIdentify

Aborts current identifying procedure started using UFM_Identify.

Syntax

```
UFM_STATUS UFM_API UFM_AbortIdentify(HUFMatcher pMatcher );
```

Parameters

- pMatcher [in]: Handle to the matcher object

Return value

UFM_STATUS

Remarks

Examples

```
UFM_STATUS      ufm_res;
HUFMatcher      hMatcher = NULL;

//Create hMatcher
//Start UFM_Identif

ufm_res = UFM_AbortIdentify(hMatcher);

if (ufm_res == UFM_OK)
{
        //UFM_AbortIdentify success
}
else
{
        //UFM_AbortIdentify failure
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_Identify
UFM_STATUS

UFM_IdentifyInit

Initializes identify with input template.

Syntax

```
UFM_STATUS UFM_API UFM_IdentifyInit(HUFMatcher pMatcher,
                                    unsigned char* pTemplate1,
                                    int nTemplate1Size);
```

Parameters

- pMatcher [in]: Handle to the matcher object
- pTemplate1 [in]: Pointer to the template
- nTemplate1Size [in]: Specifies the size of the template

Return value

UFM_STATUS

Remarks

Examples

```
//Assume template size is 1024 bytes
#define MAX_TEMPLATE_SIZE 1024

UFM_STATUS     ufm_res;
HUFMatcher     hMatcher = NULL;
unsigned char  Template1[MAX_TEMPLATE_SIZE];
int            nTemplate1Size;

//Create hMatcher
//Get Template1

ufm_res = UFM_IdentifyInit(hMatcher, Template1, nTemplate1Size);
if ((ufm_res == UFM_OK)
{
        //UFM_IdentifyInit success
}
else
{
        //UFM_IdentifyInit failure
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_IdentifyNext

Matches one input template to the template specified in UFM_IdentifyInit.

Syntax

```
UFM_STATUS UFM_API UFM_IdentifyNext(HUFMatcher pMatcher,
                                    unsigned char* pTemplate2,
                                    int nTemplate2Size,
                                    int* bIdentifySucceed );
```

Parameters

- pMatcher [in]: Handle to the matcher object
- pTemplate2 [in]: Pointer to the template array
- nTemplate2Size [in]: Specifies the size of the template array
- bIdentifySucceed [out]: Receives whether identification is succeed;
  1: identification is succeed, 0: identification is failed

Return value

UFM_STATUS

Remarks

Examples

```
#define MAX_TEMPLATE_NUM 50

UFM_STATUS     ufm_res;
HUFMatcher     hMatcher = NULL;
unsigned char  *Template2[MAX_TEMPLATE_NUM * 2];
int            nTemplate2Size[MAX_TEMPLATE_NUM * 2];
int            nTemplate2Num;
int            bIdentifySucceed;
int            i;

//Create hMatcher
//Execute UFM_IdentifyInit with query template
//Get number of templates in DB or something, and save it to nTemplate2Num

bIdentifySucceed = ;
for (i = ; i < nTemplate2Num; i++)
{
        //Get one template in DB or something, and save it to Template2 and nTemplate2Size
        ufm_res = UFM_IdentifyNext(hMatcher, Template2[i],
                                        nTemplate2Size[i], &bIdentifySucceed);
        if (ufm_res == UFM_OK)
        {
                //UFM_IdentifyNext success
        }
        else
        {
                //UFM_IdentifyNext is failed

                //Use UFM_GetErrorString function to show error string

                //return;

        }

        if (bIdentifySucceed)

        {

                //Identification is succeed break;

        }

}

if (!bIdentifySucceed)

{

        //Identification is failed

}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_IdentifyInit
UFM_STATUS

UFM_RotateTemplate

Rotates the specified template to the amount of 180 degrees.

Syntax

```
UFM_STATUS UFM_API UFM_RotateTemplate(HUFMatcher pMatcher,
                                      unsigned char* pTemplate,
                                      int nTemplateSize );
```

Parameters

- pMatcher[in]: Handle to the matcher object
- pTemplate [in / out]: Pointer to the template
- nTemplateSize [in]: Specifies the size of the template

Return value

UFM_STATUS

Remarks

Examples

```
//Assume template size if 1024 bytes
#define MAX_TEMPLATE_SIZE 1024

UFM_STATUS     ufm_res;
HUFMatcher      hMatcher = NULL;
unsigned char   Template[MAX_TEMPLATE_SIZE];
int             nTemplateSize;

//Create hMatcher
//Get a template, and save it to Template and nTemplateSize

ufm_res = UFM_RotateTemplate(hMatcher, Template, nTemplateSize);

if (ufm_res == UFM_OK)
{
        //UFM_RotateTemplate success
}
else
{
        //UFM_RotateTemplate failure
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_GetErrorString

Gets the error string for specified UFM_STATUS value.

Syntax

```
UFM_STATUS UFM_API UFM_GetErrorString(UFM_STATUS res,
                                      char* szErrorString );
```

Parameters

- res [in]: Status return value
- szErrorString [out]: Receives error string

Return value

UFM_STATUS

Remarks

Examples

```
UFS_STATUS      usf_res;
char            strError[128];

ufm_res ufm_res = UFM_GetErrorString(ufm_res, strError);
if (ufm_res == UFM_OK)
{
        //UFM_GetErrorString success
}
else
{
        //UFM_GetErrorString failure
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_GetTemplateType

Gets the parameter value.

Syntax

```
UFM_STATUS UFM_API UFM_GetTemplateType(HUFMatcher pMatcher,
                                       int* nTemplateType );
```

Parameters

- pMatcher [in]: Handle to the matcher object
- nTemplateType [out]: Receives parameter value of specified parameter type; pValue must point to adequate storage type matched to template type

Template type	Code	Description
UFM_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFM_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFM_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFM_STATUS

Remarks

Examples

```
UFM_STATUS     ufm_res;
HUFMatcher     hMatcher = NULL;
int            nTemplateType;

ufm_res = UFM_GetTemplateType(hMatcher,&nTemplateType);
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_SetTemplateType

Gets parameter value.

Syntax

```
UFM_STATUS UFM_API UFM_SetTemplateType(HUFMatcher pMatcher,
                                       int nTemplateType );
```

Parameters

- pMatcher [in]: Handle to the matcher object
- nTemplateType [in]: Parameter type; one of template type

Template type	Code	Description
UFM_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFM_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFM_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFM_STATUS

Remarks

Examples

```
UFM_STATUS     ufm_res;
HUFMatcher     hMatcher = NULL;
int            nTemplateType;

nTemplateType = UFM_TEMPLATE_TYPE_SUPREMA};

ufs_res = UFM_SetTemplateType(hMatcher, nTemplateType);
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_STATUS

Every function in a UFMatcher module returns UFM_OK when it succeeds. When it fails, it returns a value corresponding to an error code. Please find the error code on the followings if you'd like to know the information about the UFM_STATUS (integer) value.

Code	Value	Description
UFM_OK	0	Success
UFM_ERROR	-1	General error
UFM_ERR_NO_LICENSE	-101	System has no license
UFM_ERR_LICENSE_NOT_MATC	-102	License does not match
UFM_ERR_LICENSE_EXPIRED	-103	License has expired
UFM_ERR_NOT_SUPPORTED	-111	This function is not supported
UFM_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFM_ERR_DATA_CORRUPTED	-113	Template data corrupted
UFM_ERR_NOT_INITIALIZED	-114	Module is not initailized
UFM_ERR_MATCH_TIMEOUT	-401	Matching is timeout
UFM_ERR_MATCH_ABORTED	-402	Matching is aborted
UFM_ERR_TEMPLATE_TYPE	-411	Template type does not match
UFM_ERR_TEMPLATE_FORMAT	-412	Template format does not match

Chapter 3. PROGRAMMING IN C#

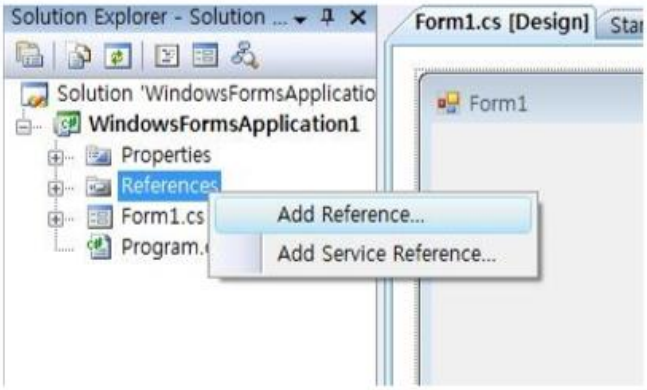
BioMini SDK provides .net Wrapper DLL for .net development.

- Suprema.UFScanner.dll
- Suprema.UFMatcher.dll

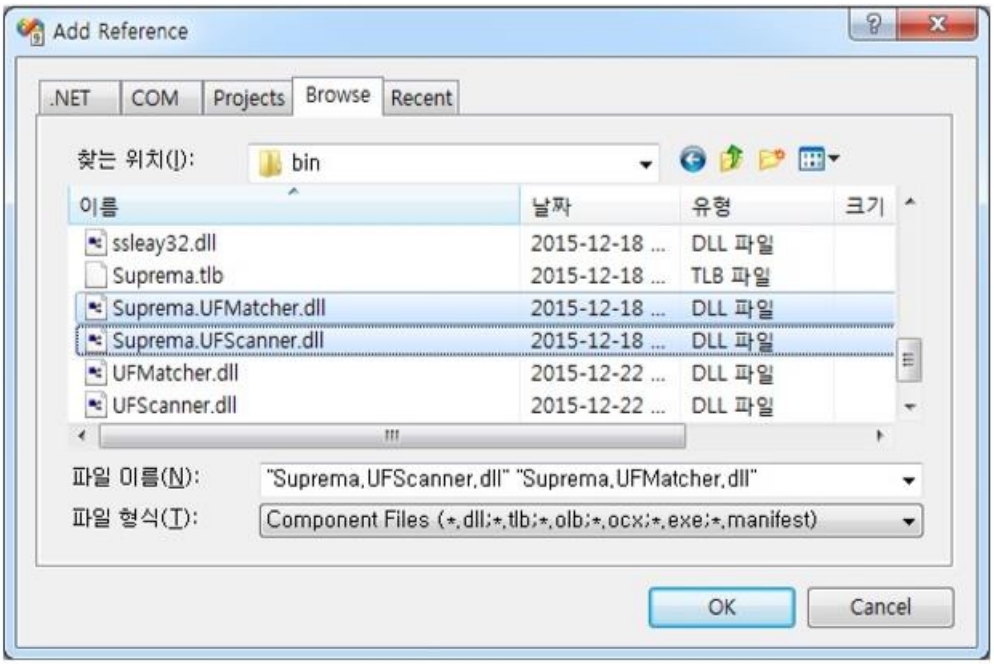
By adding above wrapper dll to the development project reference, allows to use .net API easy. To run the program, not only the .net wrapper dll, UFScanner.dll & UFMatcher.dll must be located at exe folder together.

3.1. SETTING ENVIRONMENT

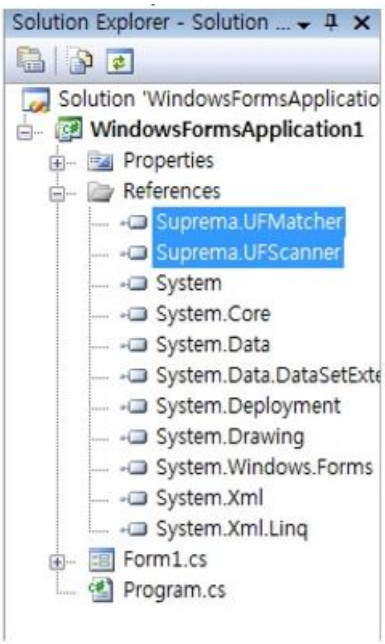
1. Click “Add Reference…” on “Solution Explorer”



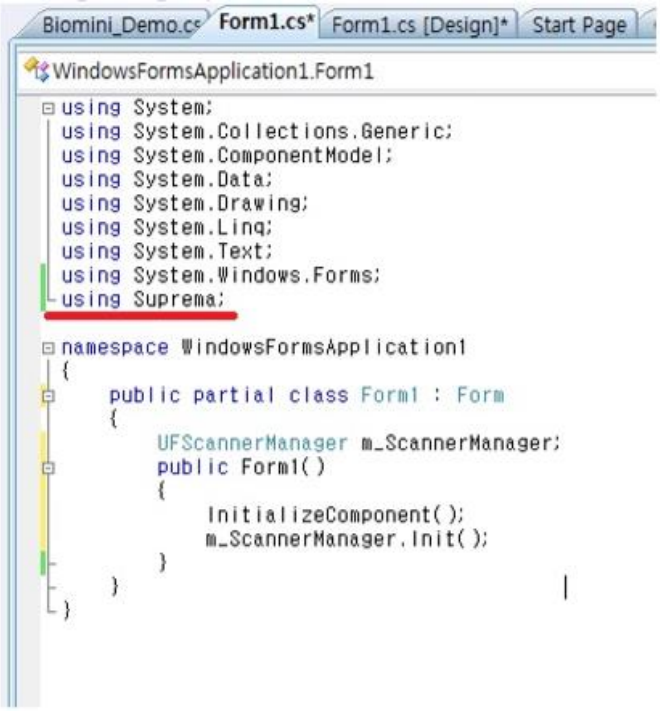
2. Select Suprema.UFMatcher.dll & Suprema.UFScanner.dll on “Browse” tab and click “OK”



3. Can be confirmed as below from “Solution Explorer”, once the reference is successfully added.



4. Can use .net APIs by adding “using Suprema;” to the source code.



3.2. API REFERENCE

UFScanner Functions

UFScannerManager

Initializes a new instance of the UFScannerManager class.

Syntax

```
public UFScannerManager(ISynchronizeInvoke synInvoke);
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScannerManager.Init

Initializes a UFScanner module.

Syntax

```
public UFS_STATUS Init();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScannerManager.Update

Enforces a UFScanner module to update the connection state of scanners.

Syntax

```
public UFS_STATUS Update();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScannerManager.Uninit

Un-initializes a UFScanner module.

Syntax

```
public UFS_STATUS Uninit()
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScannerManager.ScannerList

Gets connected scanners as UFScannerManager.ScannerList.

Syntax

```
public UFScannerManager.ScannerList Scanners { get; }
```

Parameters

- Count [out]: Gets the number of connected scanners
- Item [out]: Gets UFScanner reference by index or handle or ID of scanner

```
public UFScanner this[int Index] { get; }
public UFScanner this[IntPtr ScannerHandle] { get; }
public UFScanner this[string ScannerID] { get; }
```

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScannerManager.UFScannerManagerScannerEventArgs

Contains data of event ScannerEvent of UFScannerManager class

Syntax

```
public class UFScannerManagerScannerEventArgs : EventArgs{public string ScannerID;
                                                          public bool SensorOn;}
```

Parameters

- ScannerID [out]: Receives ID of the scanner which is occurred this event
- SensorOn [out]: true: scanner is connected, false: scanner is disconnected

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

ScannerEvent
UFS_STATUS

UFScannerManager.UFScannerCaptureEventArgs

Contains data of event CaptureEvent of UFScanner class

Syntax

```
public class UFScannerCaptureEventArgs : EventArgs{public Bitmap ImageFrame;
                                                    public int Resolution;
                                                    public bool FingerOn;}
```

Parameters

- *ImageFrame [out: Receives a captured image*
- *Resolution [out]: Receives the resolution of ImageFrame*
- *FingerOn [out]: true: finger is on the scanner, false: finger is not on the scanner*

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

CaptureEvent
UFS_STATUS

UFScanner

Variables	Description	Default value
ID	Scanner ID (get only)	-
Timeout	Timeout Timeout (millisecond unit) (: infinite)	5000
Brightness	Brightness (0 - 255); Higher value means darker image. * Not supported Device: BioMini-Slim(SFU-S20, SFU-S20B) BioMini(SFU-300)	100
Sensitivity	Sensitivity Sensitivity (0 - 7); Higher value means more sensitive	4
Serial	Scanner serial (get only)	-
SdkVersion	SDK Version (get only)	
SdkCopyright	SDK Copyright (get only)	
DetectCore	Detect core when extracting template (0: not use core, 1: use core)	0
Template Size	Template size (byte unit) (256 - 1024, 32 bytes step size)	1024
UseSIF	Use SIF(biometric data standard interchange format) (0: not use SIF, 1: use SIF)	0
DetectFake	Use live Finger Detection (0: not use LFD, 1-5: use LFD); Higher value means more strong to fake finger * Supported Device: BioMini Slim(SFU 520)	0
CompanyID	Company ID of a connected device (get only)	-
ScannerType	Scanner type of the scanner (get only)	
Handle	Scanner handle	
IsCapturing	Check if the specified scanner is running to capture which is started by UFScanner.CaptureSingleImage or UFScanner.StartCapturing (1: the capture is running, 0: the capture is not running)	
IsFingeron	Check whether a finger is placed on a scanner or not (1: a finger is on the scanner, 0: a finger is not on the scanner)	
IsSensoron	Check whether a scanner is connected or not (1: the scanner is connected, 0: the scanner is disconnected)	
Language	Language selection at runtime of Enrollui	
LFDFile	Specify the path of the engine file for upgrading the LFD Engine. Since the default engine is built into the SDK, you do not need to call it unless it is for upgrade purposes. * Supported device : BioMini Plus 2 (SFR550), BioMini Slim (SFU-520)	
LFDType	LFD operation options (O: DEFAULT, 1: ADVANCED; LFD checking level is upgrade so it can be expected improved defense performance againt to some counterfeit fingerprints.) *Supported device : BioMini Slim (SFU-520)	0
NFIQ2File	Specify the path of the NFIQ 2.0 Engine file - The NFIQ2.0 Library file in the BioMini PC SDK package is "NFIQ2.dll". (If not set, you cannot use NFIQ2.0.)	-
FPQuality	Quality Score Type (0: Suprema Quality, 1: NFIQ1.0;5 levels of valueMode between 20 and 100, 2: NFIQ1.0;5 levels of value between 1 and 5, 3: NFIQ2.0;value range from 0 and 100)	0

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.RemoveScannerCallback

Removes the registered scanner callback function.

Syntax

```
public UFS_STATUS RemoveScannerCallback();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.CaptureSingleImage

Captures single image. Captured image is stored to the internal buffer.

Syntax

```
public UFS_STATUS CaptureSingleImage();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.StartCapturing

Starts capturing. The capture is going on until the specified time exceeds.

Syntax

```
public UFS_STATUS StartCapturing();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.StartAutoCapture

Starts the automatic capture. Currently this function is working for SFR600(BioMini Slim) and SFR700(BioMini Slim 2).

Syntax

```
public UFS_STATUS StartAutoCapture();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.IsCapturing

Checks if the specified scanner is running to capture which is started by UFScanner.CaptureSingleImage() or UFScanner.StartCapturing().

Syntax

Parameters

- pbCapturing [out]: Checks if the specified scanner is running capturing;
  1: the capture is running, 0: the capture is not running

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.CaptureSingleImage()
UFScanner.StartCapturing()
UFS_STATUS

UFScanner.AbortCapturing

Aborts capturing which is started by UFScanner.CaptureSingleImage() or UFScanner.StartCapturing().

Syntax

```
public UFS_STATUS AbortCapturing();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.CaptureSingleImage()
UFScanner.StartCapturing().
UFS_STATUS

UFScanner.Extract

Extracts a template from the stored image buffer which is acquired using UFScanner.CaptureSingleImage() or UFScanner.StartCapturing().

Syntax

```
public UFS_STATUS Extract(byte[] Template,
                          out int TemplateSize,
                          out int EnrollQuality );
```

Parameters

- pTemplate [out]: Pointer to the template array; The array must be allocated in advance
- pnTemplateSize [out]: Receives the size (in bytes) of pTemplate
- pnEnrollQuality [out]: Receives the quality of enrollment; Quality value ranges from 1 to 100. Typically, this value should be above 30 for further processing such as enroll and matching. Especially in case of enrollment, the use of good quality image (above 50) is highly recommended.

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.ExtractEx

Extracts a template from the stored image buffer which is acquired using UFScanner.CaptureSingleImage() or UFScanner.StartCapturing().
This is extended version of UFScanner.extract() function to accommodate a template with large size.

Syntax

```
public UFS_STATUS ExtractEx(int nBufferSize,
                            byte[] Template,
                            out int TemplateSize,
                            out int EnrollQuality );
```

Parameters

- nBufferSize [in]: Template buffer size
- pTemplate [out]: Pointer to the template array; The array must be allocated in advance
- pnTemplateSize [out]: Receives the size (in bytes) of pTemplate
- pnEnrollQuality [out]: Receives the quality of enrollment; Quality value ranges from 1 to 100. Typically, this value should be above 30 for further processing such as enroll and matching. Especially in case of enrollment, the use of good quality image (above 50) is highly recommended.

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.CaptureSingleImage()
UFScanner.StartCapturing().
UFS_STATUS

UFScanner.SetEncryptionKey

Sets encryption key.

Syntax

```
public UFS_STATUS SetEncryptionKey(byte[] Key);
```

Parameters

pKey[out]: Pointer to the 32 bytes key array; default key is first byte is 1 and second to 32th byte are all 0

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.SetEncryptionKeyEx

Sets encryption key for standard AES algorithm.

Syntax

```
public UFS_STATUS SetEncryptionKeyEx(int AESBlockMode,
                                     byte[] Key,
                                     byte[] InitialVector);
```

Parameters

AESBlockMode [in] : Parameter type; one of AES algorithm

AES algorithm	Code	Description
UFS_ENC_MODE_DEFAULT	0	Old fashion
UFS_ENC_MODE_AES_ECB	1	Electronic Codebook
UFS_ENC_MODE_AES_CBC	2	Cipher Block Chaining
UFS_ENC_MODE_AES_PCBC	3	Propagating Cipher Block Chaining
UFS_ENC_MODE_AES_CRT	4	Cipher Feedback
UFS_ENC_MODE_AES_OFB	5	Output Feedback
UFS_ENC_MODE_AES_CFB	6	Counter

- Key[out]: Pointer to the 32bytes key array;
  default key is first byte is 1 and second to 32th byte are all 0
- InitialVector [in]: Pointer to the 16bytes initial vector (set NULL if not needed).
  UFS_ENC_MODE_AES_ECB and UFS_ENC_MODE_DEFAULT modes don't need the initial vector

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.EncryptTemplate

Encrypts template.

Syntax

```
public UFS_STATUS EncryptTemplate(byte[] TemplateInput,
                                  int TemplateInputSize,
                                  byte[] TemplateOutput,
                                  ref int TemplateOutputSize );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplate [in]: Pointer to input template data
- pTemplateInputSize [in]: Input template size
- pTemplateOutput [out]: Pointer to encrypted template data
- pnTemplateOutputSize [in / out]: Inputs allocated size of encrypted template data; Receives output template size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.DecryptTemplate

Decrypts template.

Syntax

```
public UFS_STATUS DecryptTemplate(byte[] TemplateInput,
                                  int TemplateInputSize,
                                  byte[] TemplateOutput,
                                  ref int TemplateOutputSize );
```

Parameters

- pTemplateInput [in]: Pointer to input template data(encrypted)
- nTemplateInputSize [in]: Input template size
- pTemplateOutput [out]: Pointer to output template data
- pnTemplateOutputSize [in / out]: Inputs allocated size of output template data; Receives output template size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.GetCaptureImageBuffer

Copies the capture image buffer to the specified image data array.

Syntax

```
public UFS_STATUS GetCaptureImageBuffer(out Bitmap bitmap,
                                        out int Resolution);
```

Parameters

- pImageData [out]: Pointer to image data array;
  The array must be allocated bigger than the size of capture image buffer in advance

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.GetCaptureImageBufferToBMPImageBuffer

Copies the capture image buffer to the specified image data of bmp format.

Syntax

```
public UFS_STATUS GetCaptureImageBufferToBMPImageBuffer(byte[] imageData,
                                                        out int nImageDataLength);
```

Parameters

- pImageData [out]: Pointer to bmp image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance
- pImageDataLength [out]: pointer to bmp image data size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.GetCaptureImageBufferTo19794_4ImageBuffer

Copies the capture image buffer to the specified image data of 19794_4 format.

Syntax

```
public UFS_STATUS GetCaptureImageBufferTo19794_4ImageBuffer(byte[] imageData,
                                                            out int nImageDataLength);
```

Parameters

- pImageData [out]: Pointer to 19794_4 format image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance
- pImageDataLength [out]: pointer to 19794_4 format image data size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.GetCaptureImageBufferToWSQImageBuffer

Copies the capture image buffer to the specified image data of the WSQ format.

Syntax

```
public UFS_STATUS GetCaptureImageBufferToWSQImageBuffer(byte[] wsqData,
                                                        out int TemplateSize,
                                                        float ratio);
```

Parameters

- ratio [in]: Compression ratio of image
- wsqData [out]: Pointer to WSQ format image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance
- wsqDataLen [out]: pointer to WSQ format image data size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.GetCaptureImageBufferToWSQImageBufferV ar

Copies the capture image buffer (cropped or expanded to the specified size)
to the target image data buffer of the WSQ format.

Syntax

```
public UFS_STATUS GetCaptureImageBufferToWSQImageBufferVar(byte[] wsqData,
                                                           out int TemplateSize,
                                                           float ratio,
                                                           int nWidth,
                                                           int nHeight );
```

Parameters

- Ratio [in]: Compression ratio of image
- wsqData [out]: Pointer to WSQ format image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance
- wsqDataLen [out]: pointer to WSQ format image data size
- nWidth [in]: Width to resize the capture image
- nHeight [in]: Height to resize the capture image

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.DecompressWSQBMP

Decompress WSQ file and save to BMP file.

Syntax

```
public UFS_STATUS DecompressWSQBMP(string wsqFileName,
                                   string bmpFileName);
```

Parameters

- wsqFile [in]: Specifies file name to get wsq data buffer
- bmpFile [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.DecompressWSQBMPMem

Decompress WSQ buffer and save to image data of bmp format.

Syntax

```
public UFS_STATUS DecompressWSQBMPMem(byte[] wsqBuffer,
                                      int wsqBufferLen,
                                      byte[] bmpBuffer,
                                      out int bmpBufferLen);
```

Parameters

- wsqBuffer [in]: Pointer to WSQ format image data
- wsqBufferLen [in]: Size of WSQ format image data
- bmpBuffer [out]: Pointer to bmp image data;
  The array must be allocated bigger than the size of capture image buffer in advance.
- bmpBufferLen [out]: pointer to bmp image data size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.DrawCaptureImageBuffer

Draws the fingerprint image which is acquired using UFScanner.CaptureSingleImage() or UFScanner.StartCapturing(). This function is not supported on java.

Syntax

```
public UFS_STATUS DrawCaptureImageBuffer(Graphics g,
                                         Rectangle rect,
                                         bool DrawCore);
```

Parameters

- hDC [in]: Handle to the DC where the fingerprint image is drawn
- nLeft [in]: Specifies the logical x-coordinate of the upper-left corner of the rectangle
- nTop [in]: Specifies the logical y-coordinate of the upper-left corner of the rectangle
- nRight [in]: Specifies the logical x-coordinate of the lower-right corner of the rectangle
- nBottom [in]: Specifies the logical y-coordinate of the lower-right corner of the rectangle
- bCore [in]: Specifies whether the core of fingerprint is drawn of not

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.CaptureSingleImage()
UFScanner.StartCapturing()
UFS_STATUS

UFScanner.DrawFeatureBuffer

Draws the fingerprint image which is acquired using UFScanner.CaptureSingleImage() or UFScanner.StartCapturing(). This function is not supported on java and should be called after the extraction from the last captured fingerprint image. If extraction is not performed from the last captured image, this function will not draw the feature in the image frame.

Syntax

```
public UFS_STATUS DrawFeatureBuffer(Graphics g,
                                    Rectangle rect,
                                    bool DrawCore);
```

Parameters

- hDC [in]: Handle to the DC where the fingerprint image is drawn
- nLeft [in]: Specifies the logical x-coordinate of the upper-left corner of the rectangle
- nTop [in]: Specifies the logical y-coordinate of the upper-left corner of the rectangle
- nRight [in]: Specifies the logical x-coordinate of the lower-right corner of the rectangle
- nBottom [in]: Specifies the logical y-coordinate of the lower-right corner of the rectangle
- bCore [in]: Specifies whether the core of fingerprint is drawn of not

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.CaptureSingleImage()
UFScanner.StartCapturing()
UFS_STATUS

UFScanner.SaveCaptureImageBufferToBMP

Saves the capture image buffer to the specified file of the bitmap format.

Syntax

```
public UFS_STATUS SaveCaptureImageBufferToBMP(string FileName );
```

Parameters

- FileName [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.SaveCaptureImageBufferToJPG

Saves the capture image buffer to the specified file of the jpg format.

Syntax

```
public UFS_STATUS SaveCaptureImageBufferToJPG(string FileName );
```

Parameters

- FileName[in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.SaveCaptureImageBufferTo19794_4

Saves the capture image buffer to the specified file of the 19794_4 format.

Syntax

```
public UFS_STATUS SaveCaptureImageBufferTo19794_4(string FileName );
```

Parameters

- szFileName [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.SaveCaptureImageBufferToWSQ

Saves the capture image buffer to the specified file of the WSQ format.

Syntax

```
public UFS_STATUS SaveCaptureImageBufferToWSQ(string FileName,
                                              float ratio);
```

Parameters

- ratio [in]: Compression ratio of image
- szFileName [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.SaveCaptureImageBufferToWSQVar

Saves the capture image buffer (cropped or expanded to the specified size) to the target file of the WSQ format.

Syntax

```
public UFS_STATUS SaveCaptureImageBufferToWSQVar(string FileName,
                                                 float ratio,
                                                 int nWidth,
                                                 int nHeight );
```

Parameters

- ratio [in]: Compression ratio of image
- szFileName [in]: Specifies file name to save image buffer
- nWidth [in]: Width to resize the capture image
- nHeight [in] : Height to resize the capture image

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.ClearCaptureImageBuffer

Clears the capture image buffer stored to the internal buffer.

Syntax

```
public UFS_STATUS ClearCaptureImageBuffer();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.GetErrorString

Gets the error string for specified UFS_STATUS() value.

Syntax

```
public static UFS_STATUS GetErrorString(UFS_STATUS res,
                                        out string ErrorString);
```

Parameters

- szErrorString [out]: Receives error string

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.SelectTemplate

Selects n number of good templates from m number of input templates.

Syntax

```
public UFS_STATUS SelectTemplate(byte[,] ppTemplateInput,
                                 int[] pnTemplateInputSize,
                                 int nTemplateInputNum,
                                 byte[,] ppTemplateOutput,
                                 int[] pnTemplateOutputSize,
                                 int nTemplateOutputNum);

public UFS_STATUS SelectTemplate(byte[][] ppTemplateInput,
                                 int[] pnTemplateInputSize,
                                 int nTemplateInputNum,
                                 byte[][] ppTemplateOutput,
                                 int[] pnTemplateOutputSize,
                                 int nTemplateOutputNum);
```

Parameters

- ppTemplateInput [in]: Array pointer to the input template arrays
- pnTemplateInputSize [in]: Array pointer to input templates' size
- nTemplateInputNum [in]: Number of input templates
- ppTemplateOutput [out]: Array pointer to the output template arrays
- pnTemplateOutputSize [out]: Array pointer to the output templates' size
- nTemplateOutputNum [in]: Number of output templates;
  should be less than input template number by more than one

Return value

UFS_STATUS

Remarks

Examples

```
const int MAX_TEMPLATE = 10;
const int MAX_TEMPLATE_SIZE = 1024;
byte[,] pInData = new byte[MAX_TEMPLATE_SIZE,MAX_TEMPLATE];
byte[,] pOutData = new byte[MAX_TEMPLATE_SIZE,MAX_TEMPLATE];
int[] pnInData = new int[MAX_TEMPLATE];
int[] pnOutData = new int[MAX_TEMPLATE];
int nInTemplate = 4;
int nOutTemplate =1;

Scanner.SelectTemplate(pInData, pnInData, nInTemplate, pOutData,
                                pnOutData, nOutTemplate);

byte[][] pInData;
byte[][] pOutData;
int[] pnInData = new int[MAX_TEMPLATE];
int[] pnOutData = new int[MAX_TEMPLATE];
int nInTemplate = 4;
int nOutTemplate =1;

Scanner.SelectTemplate(pInData, pnInData, nInTemplate, pOutData, pnOutData, nOutTemplate);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.SelectTemplateEx

Selects n number of good templates from m number of input templates
This is extended version of UFScanner.SelectTemplate() function to accommodate the template with large size.

Syntax

```
public UFS_STATUS SelectTemplateEx(int nBufferSize,
                                   byte[,] ppTemplateInput,
                                   int[] pnTemplateInputSize,
                                   int nTemplateInputNum,
                                   byte[,] ppTemplateOutput,
                                   int[] pnTemplateOutputSize,
                                   int nTemplateOutputNum);

public UFS_STATUS SelectTemplateEx(int nBufferSize
                                   byte[][] ppTemplateInput,
                                   int[] pnTemplateInputSize,
                                   int nTemplateInputNum,
                                   byte[][] ppTemplateOutput,
                                   int[] pnTemplateOutputSize,
                                   int nTemplateOutputNum);
```

Parameters

- nBufferSize [in]: Template buffer size
- ppTemplateInput [in]: Array pointer to the input template arrays
- pnTemplateInputSize [in]: Array pointer to the input templates' size
- nTemplateInputNum [in]: Number of input templates
- ppTemplateOutput [out]: Array pointer to the output template arrays
- pnTemplateOutputSize [out]: Array pointer to the output templates' size
- nTemplateOutputNum [in]: Number of output templates;
  should be less than input template number by more than one

Return value

UFS_STATUS

Remarks

Examples

```
const int MAX_TEMPLATE = 10;
const int MAX_TEMPLATE_SIZE = 1024;

byte[,] pInData = new byte[MAX_TEMPLATE_SIZE,MAX_TEMPLATE];
byte[,] pOutData = new byte[MAX_TEMPLATE_SIZE,MAX_TEMPLATE];
int[] pnInData = new int[MAX_TEMPLATE];
int[] pnOutData = new int[MAX_TEMPLATE];
int nInTemplate = 4;
int nOutTemplate =1;

Scanner.SelectTemplateEx(384 , pInData, pnInData, nInTemplate, pOutData,
                                    pnOutData, nOutTemplate);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.SelectTemplate()
UFS_STATUS

UFScanner.GetFPQuality

Calculates the quality score of an image according to FPQualityMode variable.

NFIQ 2.0 (NIST Fingerprint Image Quality) is the quality score of an image as defined in the draft.
To use NFIQ 2.0, you need to set variables as follows.
1) Set FPQualityMode variable to "3".
2) Set NFIQ2File variable to the path where the NFIQ 2.0 Library file is located.
(For example, the NFIQ2.0 Library file in the BioMini SDK package is NFIQ2.dll.)

Syntax

```
public UFS_STATUS GetFPQuality(byte[] fpImageData,
                               int nWidth,
                               int nHeight,
                               out int nFPQuality);
```

Parameters

- fpImageData[in]: Raw capture image data
- nWidth [in]: Width of capture image data
- nHeight [in]: Height of capture image data
- nFPQuality[in]: NIST quality score of image data

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

FPQualityMode
UFS_STATUS

UFScanner.GetFeatureNumber

Get the number of Minutiae from the template data.

Syntax

```
public UFS_STATUS GetFeatureNumber(byte[] Template,
                                   int TemplateSize,
                                   out int FeatureNumber);
```

Parameters

- pTemplate [in]: Template data
- nTemplateSize [in]: Size of template data
- pnFeatureNum [out]: The number of minutiae from pTemplate

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.EnrollUI

Generate the fingerprint enrollment dialog. This function can be called after executing UFScanner.Init.
Enrolling a fingerprint is extracting a template from finger and saving the template. Below sample's UFS_EnrollUI function captures a fingerprint image after setting the template type. And extracts a template from captured fingerprint image. The extracted template will be saved in a specific template array, which is a parameter of the UFS_EnrollUI function. It supported only for Windows environment.

*Constraints
– You should have 'img' folder to use graphical backgrounds and buttons. The application uses the img folder should be at the upper level folder. For example, if the application is at the /bin/sample, 'img' folder should be at the location of /bin/sample/img.
– Enrollment UI is based on COM interface. Thus you should register dll file before use. You can use the pre-coded script (register_enrollui.bat) to register the dll file, or simple type the command 'regsvr32.exe IEnrollUI.dll' at the command prompt.

Syntax

```
public Suprema.UFS_STATUS EnrollUI(int nTimeout,
                                   int nOptions,
                                   byte[] pUF_FIR_Buf,
                                   int[] pUF_FIR_Buf_Len,
                                   byte[] pISO_FIR_Buf,
                                   int[] pISO_FIR_Buf_Len,
                                   byte[] pImages_Path,
                                   byte[] pImages_buf,
                                   int[] pImages_Buf_Len);

public UFS_STATUS EnrollUI(int nTimeout,
                           int nOptions,
                           byte[] pUF_FIR_Buf,
                           int[] pUF_FIR_Buf_Len,
                           byte[] pISO_FIR_Buf,
                           int[] pISO_FIR_Buf_Len,
                           byte[] pImages_Path);
```

```
public UFS_STATUS EnrollUI(int nTimeout,
                           int nOptions,
                           out List<byte[]> suprema_templates,
                           out List<byte[]> iso_templates,
                           string out_image_dir,
                           out List<byte[]> images);

public UFS_STATUS EnrollUI(int nTimeout,
                           int nOptions,
                           out List<byte[]> suprema_templates,
                           out List<byte[]> iso_templates,
                           out List<byte[]> images);

public UFS_STATUS EnrollUI(int nTimeout,
                           int nOptions,
                           out List<byte[]> suprema_templates,
                           out List<byte[]> iso_templates);

public UFS_STATUS EnrollUI(int nTimeout,
                           int nOptions,
                           out List<byte[]> suprema_templates,
                           out List<byte[]> iso_templates ,
                           string out_image_dir)
```

Parameters

- nTimeout [in]: Timeout of the capture
- nOptions [in]: Options for enrollment. Matching level, image Quality, number of fingerprints for enrollment, number of templates per finger
- pUF_FIR_Buf [out]: Pointer to the byte array for suprema template. This data pointer is assigned by maximum 1024*20
- pUF_FIR_Buf_Len [out]: Pointer to the int array for length of suprema template buffer
- pISO_FIR_Buf [out]: Pointer to the byte array for ISO template.
  This data pointer is assigned by maximum 1024 * 20

- pISO_FIR_Buf_Len [out]: Pointer to the int array for length of ISO template buffer
- pImages_Path [in]: Path to captured images to be saved. If NULL value is passed, nothing will be saved
- pImage_Buf [out]: Pointer to the byte array for image buffer. This data pointer is assigned by maximum 320 * 480
- List<byte[]> suprema_templates [out]: List of templates with Suprema format
- List<byte[]> iso_templates [out]: List of templates with ISO 19794-2 format
- out_image_dir [in]: Path to captured images to be saved
- List<byte[]> images [out]: List of image buffers

Return value

UFS_STATUS

Remarks

Examples

```
int nTimeout = 5000; UFS_STAT
US nRet = Scanner.EnrollUI(cbTimeout.SelectedIndex, nOptions, pUFBuf,
                                    pUFBufSize, pISOBuf, pISOBufSize , "Image_path");

int nTimeout = 5000;
List<byte[]> pUFList = new List<byte[]>();
List<byte[]> pISOList = new List<byte[]>();
List<byte[]> pImages = new List<byte[]>();
UFS_STATUS nRet = Scanner.EnrollUI(nTimeout, nOptions, out pUFList,
                                          out pISOList, "Image_path",out pImages);
int nTimeout = 5000;
List<byte[]> pUFList = new List<byte[]>();
List<byte[]> pISOList = new List<byte[]>();
List<byte[]> pImages = new List<byte[]>();
UFS_STATUS nRet = Scanner.EnrollUI(nTimeout, nOptions, out pUFList, out pISOList, out pImages);

int nTimeout = 5000;
List<byte[]> pUFList = new List<byte[]>();
List<byte[]> pISOList = new List<byte[]>();
UFS_STATUS nRet = Scanner.EnrollUI(nTimeout, nOptions, out pUFList, out pISOList);

int nTimeout = 5000;
List<byte[]> pUFList = new List<byte[]>();
List<byte[]> pISOList = new List<byte[]>();
UFS_STATUS nRet = Scanner.EnrollUI(nTimeout, nOptions, out pUFList, out pISOList, "Image_path");
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.VerifyUI

Generate the fingerprint verification dialog. This function can be called after executing UFScanner.Init and UFScanner.EnrollUI. Two fingerprints can be verified whether they are matched or not. Below sample's UFS_VerifyUI function captures a fingerprint image and extracts a template from the image. And execute 1:1 matching using extracted template and templates enrolled from UFScanner.EnrollUI.

*Constraints
– Refer to the UFScanner.EnrollUI constraints.

Syntax

```
public Suprema.UFS_STATUS VerifyUI(int nTimeout,
                                   int nOptions,
                                   int nFPTemplateType,
                                   byte[] pFIR_Buf,
                                   int[] pFIR_Buf_Len,
                                   byte[] pImage_Name,
                                   out int nFingerIndex);
public UFS_STATUS VerifyUI(int nTimeout,
                           int nOptions,
                           int nFPTemplateType,
                           List<byte[]> templates_to_test,
                           string out_image_dir,
                           out int nFingerIndex);
```

Parameters

- nTimeout [in]: Timeout of the capture
- nOptions [in]: Options for enrollment. Matching level, image Quality, number of fingerprints for enrollment, number of templates per finger
- nFPTemplateType [in]: Template type for matching enrolled templates with captured fingerprint
- pFIR_Buf [out]: Pointer to the byte array for template
- pFIR_Buf_Len [out]: Pointer to the int array for length of template buffer
- pImage_Name[in]:: Path to captured images to be saved
- nFingerIndex [in]: Matched finger index from enrolled templates.
- If this value is -1, the matching result is failed List<byte[]> templates_to_test [out]: List of template buffers
- out_image_dir [in]: Path to captured images to be save

Return value

UFS_STATUS

Remarks

Examples

```
byte[] m_pUFTemplateBuf;
int[] m_pUFTemplateBuddfSize;
int nFingerIndex;

nRet = Scanner.VerifyUI(nTimeout, nOptions, 2001/*Template Type */,
                        m_pUFTemplateBuf, m_pUFTemplateBufSize, "./Image_path", out nFingerIndex);

List<byte[]> m_pUFTemplateBuf;
int nFingerIndex;

nRet = Scanner.VerifyUI(nTimeout, nOptions, 2001/*Template Type */,
                                  m_pUFTemplateBuf, "./Image_path", out nFingerIndex);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.EnrollUI
UFS_STATUS

UFScanner.CaptureSingleUI

Performs same as UFScanner.CaptureSingleImage and Popup Window appears once the capturing starts to show a captured image then disappears.

Syntax

```
public UFS_STATUS CaptureSingleUI(int nTimeout,
                                  int nOptions,
                                  byte[] pImage_buf,
                                  out int nImage_width,
                                  out int nImage_height,
                                  byte[] pImages_Path,
                                  out int nFingerQ);


public UFS_STATUS CaptureSingleUI(int nTimeout,
                                  int nOptions,
                                  out Bitmap capturedImage,
                                  string pImages_Path,
                                  out int nFingerQ);
```

Parameters

- nTimeout [in]: Applicable timeout parameter for capture single function
- nOptions [in]: Same as UFScanner.EnrollUI option
- pImage_buf [out]: Buffer of a captured image (The memory buffer has to me managed by the user)
- nImage_width, nImage_height [out]: Width / Height of a captured image (pixel)
- pImages_Path [in]: Path to captured images to be saved
- nFingerQ [out]: Returns same score as UFscanner.GetFPQuality

Return value

UFS_STATUS

Remarks

Examples

```
int nTimeout = 5000;
Bitmap bitmap;
int quality=; UFS_STATUS = Scanner.CaptureSingleUI(nTimeout, nOptions, out bitmap,
                                                   "./Image_path", out quality);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.CaptureSingleImage
UFscanner.GetFPQuality
UFS_STATUS

UFScanner.CaptureSingleUIEx

Performs same as UFScanner.CaptureSingleUI but it cannot specify the path to save an image.

Syntax

```
public UFS_STATUS CaptureSingleUIEx(int nTimeout,
                                    int nOptions,
                                    byte[] pCapture_buf,
                                    int[] pnCaptured_Buf_Len,
                                    out int nFingerQ);
```

Parameters

- nTimeout [in]: Applicable timeout parameter for capture single function (msec)
- nOptions [in]: Same as UFScanner.EnrollUI option
- pCapture_buf [in]: Buffer of a captured image (The memory buffer has to me managed by the user)
- pnCaptured_Buf_Len [in]: Size of a captured image (pixel)
- nFingerQ [in]: Returns same score as UFScanner.GetFPQuality

Return value

UFS_STATUS

Remarks

Examples

```
int nTimeout = 5000;
byte[] pbCapturedImageBuf    = new byte[MAX_TEMPLATE_SIZE*10];
int[] pnCapturedImageLenBuf = new int[10];
int nFQuality = ;

Scanner.CaptureSingleUIEx(nTimeout, nOptions, pbCapturedImageBuf,
                          pnCapturedImageLenBuf, nFQuality);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFScanner.CaptureSingleUI
UFScanner.EnrollUI

UFScanner.GetFPQuality
UFS_STATUS

UFScanner.SetScanner

Sets current scanner instance using index / handle / ID information

Syntax

```
public UFS_STATUS SetScanner(int ScannerIndex);


public UFS_STATUS SetScanner(string ScannerID);


public UFS_STATUS SetScanner(IntPtr ScannerHandle);
```

Parameters

- ScannerIndex [out]: Index of a scanner
- ScannerID [out]: Scanner ID
- ScannerHandle [out]: Handler of a scanner

Return value

UFS_STATUS

Remarks

Examples

```
private ArrayList m_LocalScannerArray;
int nScannerNumber;
int i;
_UFScanner.UFS_GetScannerNumber(out nScannerNumber);
for (i = ; i < nScannerNumber; i++){
            m_LocalScannerArray.Add(new UFScanner());
            ((UFScanner)m_LocalScannerArray[i]).SetScanner(i);
}
private ArrayList m_LocalScannerArray;
int nScannerNumber;
int i;
_UFScanner.UFS_GetScannerNumber(out nScannerNumber);
for (i = ; i < nScannerNumber; i++){
            m_LocalScannerArray.Add(new UFScanner());
            ((UFScanner)m_LocalScannerArray[i]).SetScanner("Scanner ID");
}
```

```
private ArrayList m_LocalScannerArray;
int nScannerNumber;
int i;
_UFScanner.UFS_GetScannerNumber(out nScannerNumber);
IntPtr hScanner;
UFS_GetScannerHandle( , out hScanner);

for (i = ; i < nScannerNumber; i++){
            m_LocalScannerArray.Add(new UFScanner());
            ((UFScanner)m_LocalScannerArray[i]).SetScanner(hScanner);
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFScanner.PackOptions

Packs following parameters into one parameter with int type (Security level, image quality, number of fingerprints for enrollment, number of templates per finger).
It needs for 'nOptions' parameter of UFScanner.CaptureSingleUI, UFScanner.EnrollUI or UFScanner.VerifyUI.

Syntax

```
public UFS_STATUS PackOptions(int securityLevel,
                              int quality,
                              int fingersToEnroll,
                              int templatesPerFinger,
                              out int options);
```

Parameters

- securityLevel[in]: Security level
- quality [in]: Image quality to determine enrolling
- fingersToEnroll[in]: Number of fingers to receive
- templatesPerFinger[in]: Number of times input is received per finger
- options[out]: Result option value

Return value

UFS_STATUS

Remarks

Examples

```
int nSecurelevel = 1;
int nEnrollQ = 90;
int nFingerToEnroll = 5;
int nTemplatePerFinger = 2;
int nOptions;

Scanner.PackOptions(nSecurelevel, nEnrollQ, nFingersToEnroll,
                    nTemplatesPerFinger, out nOptions);
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_TEMPLATE_TYPE

Sets or Gets the template type. Please find the template type on the followings if you'd like to know the information about the UFS_TEMPLATE_TYPE (Enum) value.

Syntax

```
public enum UFS_TEMPLATE_TYPE : int
```

Parameters

Template type	Code	Description
UFS_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFS_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFS_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SCANNER_TYPE

Gets the scanner type that is assigned to the scanner handle. Please find the scanner type on the followings if you'd like to know the information about the UFS_SCANNER_TYPE Enumeration value.

Syntax

```
public enum UFS_SCANNER_TYPE : int
```

Parameters

Scanner type	Code	Description
SFR200	1001	Suprema SFR200
SFR300	1002	Suprema SFR300-S
SFR300v2	1003	Suprema SFR300v2, SFR400, SFR410
SFR410	1004	BioMini OC4
SFR500	1005	BioMini Plus
SFR600	1006	BioMini Slim
SFR550	1007	BioMini Plus 2
SFR700	1008	BioMini Slim 2
BMS3	1012	BioMini Slim 3

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SCANNER_PROC

Defines the delegate for the scanner event ScannerEvent

Syntax

```
public delegate void UFS_SCANNER_PROC(object sender,
                                      UFScannerManagerScannerEventArgs e);
```

Parameters

- sender [in]: The sender of the event
- e [in]: A UFScannerManagerScannerEventArgs that contains the event data

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

ScannerEvent
UFS_STATUS

UFS_CAPTURE_PROC

Defines the delegate for the capture event CaptureEvent

Syntax

```
public delegate void UFS_CAPTURE_PROC(object sender,
                                      UFScannerCaptureEventArgs e);
```

Parameters

- sender [in]: The sender of the event
- e [in]: A UFScannerCaptureEventArgs that contains the event data

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

CaptureEvent
UFS_STATUS

ScannerEvent

Occurs when the scanner is connected or disconnected.

Syntax

```
public event UFS_SCANNER_PROC ScannerEvent;
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

CaptureEvent

After a capturing is started using StartCapturing,
this event occurs when an image frame is captured from the scanner

Syntax

```
public event UFS_CAPTURE_PROC CaptureEvent;
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_STATUS

Every function in a UFScanner module returns OK when it succeeds. When it fails, it returns a value corresponding to an error code. Please find the error code on the followings if you'd like to know the information about the UFS_STATUS (Enum) value.

Code	Value	Description
UFS_OK	0	Success
UFS_ERROR	-1	General error
UFS_ERR_DEVICE_NOT_RESPOND	-10	Device is not responded
UFS_ERR_CAPTURE_TIMEOUT	-11	The time for capturing exceeds
UFS_ERR_USB_TIMEOUT	-12	The time for USB transfer exceeds
UFS_ERR_NO_LICENSE	-101	Device is not connected or License is not located
UFS_ERR_LICENSE_NOT_MATCH	-102	License does not match
UFS_ERR_LICENSE_EXPIRED	-103	License has expired
UFS_ERR_NOT_SUPPORTED	-111	This function is not supported
UFS_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFS_ERR_UNKNOWN_TEMPLATE_FORMAT	-120	Abnormal template recognized
UFS_ERR_INVALIDE_ENCRYPTION	-121	Template buffer, encrypted template size is incorrect
UFS_ERR_INVALID_MODE	-122	Device is sleep mode
UFS_ERR_ALREADY_INITIALIZED	-201	Module is already initialized
UFS_ERR_NOT_INITIALIZED	-202	Module is not initialized
UFS_ERR_DEVICE_NUMBER_EXCEED	-203	Device number exceeds
UFS_ERR_LOAD_SCANNER_LIBRARY	-204	Error on loading the library of a scanner
UFS_ERR_CAPTURE_RUNNING	-211	Capturing is started using UFS_CaptureSingleImage of UFS_StartCapturing
UFS_ERR_CAPTURE_FAILED	-212	Capturing is timeout or aborted
UFS_ERR_FAKE_FINGER	-221	Fake finger is detected
UFS_ERR_FINGER_ON_SENSOR	-231	Remove finger from sensor
UFS_ERR_TIMEOUT	-241	Time out
UFS_ERR_NOT_GOOD_IMAGE	-301	Input image is not good
UFS_ERR_EXTRACTION_FAILED	-302	Extraction is failed
UFS_ERR_ID_NOT_FOUND	-500	ID not found on device
UFS_ERR_NO_MATCH_FOUND	-501	Not registered fingerprint
UFS_ERR_NOT_IDENTICAL	-502	Not matched
UFS_ERR_CANNOT_ENROLL	-510	Cannot enroll fingerprint
UFS_ERR_NO_DATA	-511	No data found on device
UFS_ERR_NOT_AUTHORIZED	-520	Not authorized to use this API

In case UFS_PARAM_DETECT_CORE is "1" (=use), you may get the following error

Code	Value	Description
UFS_ERR_CORE_NOT_DETECTED	-351	Core is not detected
UFS_ERR_CORE_TO_LEFT	-352	Move finger to left
UFS_ERR_CORE_TO_LEFT_TOP	-353	Move finger to left-top
UFS_ERR_CORE_TO_TOP	-354	Move finger to top
UFS_ERR_CORE_TO_RIGHT_TOP	-355	Move finger to right-top
UFS_ERR_CORE_TO_RIGHT	-356	Move finger to right
UFS_ERR_CORE_TO_RIGHT_BOTTOM	-357	Move finger to right-bottom
UFS_ERR_CORE_TO_BOTTOM	-358	Move finger to bottom
UFS_ERR_CORE_TO_LEFT_BOTTOM	-359	Move finger to left-bottom
UFS_ERR_FINGER_TOO_RIGHT	-401	Finger is too close to the right edge
UFS_ERR_FINGER_TOO_LEFT	-402	Finger is too close to the left edge
UFS_ERR_FINGER_TOO_TOP	-403	Finger is too close to the top
UFS_ERR_FINGER_TOO_BOTTOM	-404	Finger is too close to the bottom
UFS_ERR_FINGER_TIP	-405	Do not put the fingertip

UFMatcher Functions

UFMatcher

Creates a matcher object.

Syntax

```
new UFMatcher();
```

Variables of a matcher

Variable	Description	Default value
InitResult	Result of initializing UFMatcher (0: not initialized, 1: initialized)	0
FAST_MODE	FAST_MODE Fast Mode (0: not use fast mode, 1: use fast mode)	1
SECURITY_LEVEL	Get the False Accept Ratio(FAR) (1: Below 1%, 2: Below 0.1%, 3: Below SECURITY_LEVEL 0.01%, 4: Below 0.001%, 5: Below 0.0001%, 6: Below 0.00001%, 7: Below 0.000001%)	4
USE_SIF	USE_SIF Use SIF (0: not use SIF, 1: use SIF)	**0**

Parameters

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFSMatcher.lib

See also

UFM_STATUS

UFMatcher.Delete

Deletes specified matcher object.

Syntax

```
public UFM_STATUS Delete();
```

Parameters

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFMatcher.Verify

Compares two extracted templates.

Syntax

```
public UFM_STATUS Verify(byte[] Template1,
                         int Template1Size,
                         byte[] Template2,
                         int Template2Size,
                         out bool VerifySucceed );
```

Parameters

- pTemplate1 [in]: Pointer to the template1
- nTemplate1Size [in]: Specifies the size of the template1
- pTemplate2 [in]: Pointer to the template2
- nTemplate2Size [in]: Specifies the size of the template2
- bCerifySucceed [out]: Receives, whether verification is succeed;
  1: verification is succeed, 0: verification is failed

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFMatcher.VerifyEx

Performs same as UFMatcher.Verify, and returns matching score by 6th parameter
(matching score in between 0~1, idle match as the score is close to 1)

Syntax

```
public UFM_STATUS Verify(byte[] Template1,
                         int Template1Size,
                         byte[] Template2,
                         int Template2Size,
                         out float fScore,
                         out bool VerifySucceed );
```

Parameters

- Template1[in]: Pointer to the Template1
- Template1Size[in]: Specifies the size of the Template1
- Template2[in]: Pointer to the Template2
- Template2Size[in]: Specifies the size of the Template2
- fScore [out]: Matching score between Template1 and Template2
- VerifySucceed[out]: Receives, whether verification is succeed; 1: verification is succeed, 0: verification is failed

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFMatcher.Verify
UFM_STATUS

UFMatcher.Identify

Compares a template with given template array.

Syntax

```
public UFM_STATUS Identify(byte[] Template1,
                           int Template1Size,
                           byte[] Template2Array,
                           int[] Template2SizeArray,
                           int Template2Num,
                           int Timeout,
                           out int MatchTemplate2Index);

public UFM_STATUS Identify(byte[] Template1,
                           int Template1Size,
                           byte[][] Template2Array,
                           int[] Template2SizeArray,
                           int Template2Num,
                           int Timeout,
                           out int MatchTemplate2Index);
```

Parameters

- Template1 [in]: Pointer to the template
- Template1Size [in]: Specifies the size of the template
- Template2Array [in]: Pointer to the template array
- Template2SizeArray [in]: Pointer to the template size array
- Template2Num [in]: Specifies the number of templates in the template array
- Timeout [in]: Specifies maximum time for identifying in milliseconds; If elapsed time for identifying exceeds nTimeout, function stops further identifying and returns UFM_ERR_MATCH_TIMEOUT; 0 means infinity
- MatchTemplate2Index [out]: Receives the index of matched template in the template array; -1 means Template1 is not matched to all of templates in Template2Array

Return value

UFM_STATUS

Remarks

Examples

```
byte[] Template = new byte[MAX_TEMPLATE_SIZE];
int TemplateSize; b
byte[][] template_all;
int[] templateSize_all;
int nMaxTemplateNum = 50;
int MatchIndex = ;

ufm_res = m_Matcher.Identify(Template, TemplateSize, template_all,
                                    templateSize_all, nMaxTemplateNum, 5000, out MatchIndex);
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFMatcher.IdentifyMT

Compares a template with given template array.
UFMatcher.IdentifyMT function uses multi threads internally for faster identifying in multi-core systems.

Syntax

```
public UFM_STATUS IdentifyMT(byte[] Template1,
                             int Template1Size,
                             byte[,] Template2Array,
                             int[] Template2SizeArray,
                             int Template2Num,
                             int Timeout,
                             out int MatchTemplate2Index);
public UFM_STATUS IdentifyMT(byte[] Template1,
                             int Template1Size,
                             byte[][] Template2Array,
                             int[] Template2SizeArray,
                             int Template2Num,
                             int Timeout,
                             out int MatchTemplate2Index);
```

Parameters

- Template1 [in]: Pointer to the template
- Template1Size [in]: Specifies the size of the template
- Template2Array [in]: Pointer to the template array
- Template2SizeArray [in]: Pointer to the template size array
- Template2Num [in]: Specifies the number of templates in the template array
- Timeout [in]: Specifies maximum time for identifying in milliseconds; If elapsed time for identifying exceeds nTimeout, function stops further identifying and returns UFM_ERR_MATCH_TIMEOUT; 0 means infinity
- MatchTemplate2Index [out]: Receives the index of matched template in the template array;
  -1 means Template1 is not matched to all of templates in Template2Array

Return value

UFM_STATUS

Remarks

Examples

```
byte[] Template = new byte[MAX_TEMPLATE_SIZE];
int TemplateSize;
byte[][] template_all;
int[] templateSize_all;
int nMaxTemplateNum = 50;
int MatchIndex = ;

ufm_res = m_Matcher.IdentifyMT(Template, TemplateSize, template_all,
                                        templateSize_all, nMaxTemplateNum, 5000, out MatchIndex)
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFMatcher.AbortIdentify

Aborts current identifying procedure started using UFMatcher.Identify() and UFMatcher.IdentifyMT().

Syntax

```
public UFM_STATUS AbortIdentify()
```

Parameters

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFMatcher.Identify()
UFMatcher.IdentifyMT().
UFM_STATUS

UFMatcher.IdentifyInit

Initializes identify with input template.

Syntax

```
public UFM_STATUS IdentifyInit(byte[] Template1,
                               int Template1Size );
```

Parameters

- pTemplate1 [in]: Pointer to the template
- nTemplate1Size [in]: Specifies the size of the template

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFMatcher.IdentifyNext

Matches one input template to the template specified in UFMatcher.IdentifyInit().

Syntax

```
public UFM_STATUS IdentifyNext(byte[] Template2,
                               int Template2Size,
                               out bool IdentifySucceed) };
```

Parameters

- pTemplate2 [in]: Pointer to the template array
- nTemplate2Size [in]: Specifies the size of the template array
- bIdentifySucceed [out]: Receives whether identification is succeed;
  1: identification is succeed, 0: identification is failed

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFMatcher.IdentifyInit()
UFM_STATUS

UFMatcher.RotateTemplate

Rotates the specified template to the amount of 180 degrees.

Syntax

```
public UFM_STATUS RotateTemplate(byte[] Template,
                                 int TemplateSize);
```

Parameters

- pTemplate [in / out]: Pointer to the template
- nTemplateSize [in]: Specifies the size of the template

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFMatcher.GetErrorString

Gets the error string for the specified UFM_STATUS() value.

Syntax

```
public static UFM_STATUS GetErrorString(UFM_STATUS res,
                                        out string ErrorString );
```

Parameters

- res [in]: Status return value
- szErrorString [out]: Receives error string

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_TEMPLATE_TYPE

Sets or Gets the template type. Please find the template type on the followings
if you'd like to know the information about the UFM_TEMPLATE_TYPE (Enum) value.

Syntax

```
public enum UFM_TEMPLATE_TYPE : int
```

Parameters

Template type	Code	Description
UFM_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFM_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFM_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_STATUS

Every function in a UFMatcher module returns UFM_OK when it succeeds. When it fails, it returns a value corresponding to an error code. Please find the error code on the followings if you'd like to know the information about the UFM_STATUS (integer) value.

Code	Value	Description
UFM_OK	0	Success
UFM_ERROR	-1	General error
UFM_ERR_NO_LICENSE	-101	System has no license
UFM_ERR_LICENSE_NOT_MATC	-102	License does not match
UFM_ERR_LICENSE_EXPIRED	-103	License has expired
UFM_ERR_NOT_SUPPORTED	-111	This function is not supported
UFM_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFM_ERR_DATA_CORRUPTED	-113	Template data corrupted
UFM_ERR_NOT_INITIALIZED	-114	Module is not initailized
UFM_ERR_MATCH_TIMEOUT	-401	Matching is timeout
UFM_ERR_MATCH_ABORTED	-402	Matching is aborted
UFM_ERR_TEMPLATE_TYPE	-411	Template type does not match
UFM_ERR_TEMPLATE_FORMAT	-412	Template format does not match

Chapter 4. PROGRAMMING IN JAVA

4.1. SETTING ENVIRONMENT

You need the following packages installed BioMini SDK, JAVA SDK 1.4 or higher,

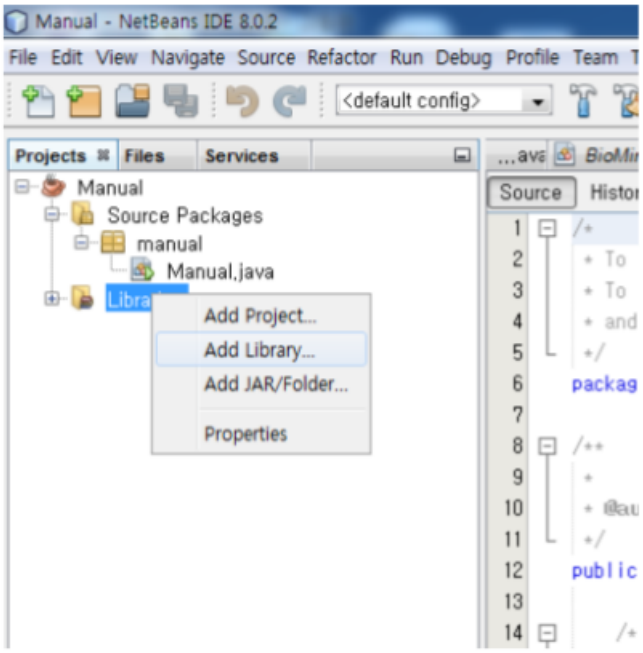
JNI package file (“BioMiniSDK.jar” is at the following location “<BioMini SDK installed path>\bin\java”)

1. **Install Java SDK**
   You should install Java SDK 1.8 or higher version.
   For more information, please see http://www.oracle.com.

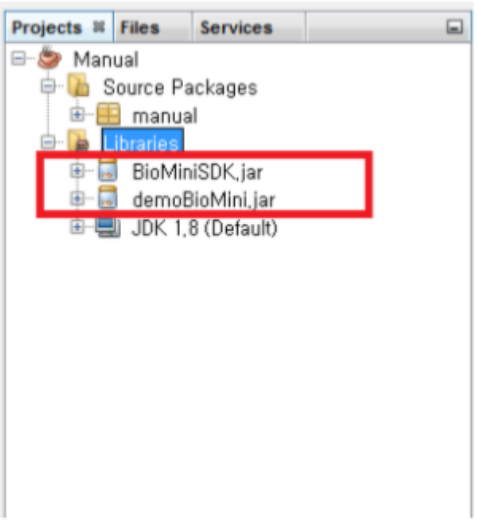
2. **Set Classpath**
   After java sdk installation, you must set classpath as following example.
   “CLASSPATH=.;<path to the BioMiniSDK.jar file>;”

3. **Build and run sample**
   You can build and run the JNI demo application in the following location.
   <BioMini SDK installed folder>\samples\java\demoBioMini.java

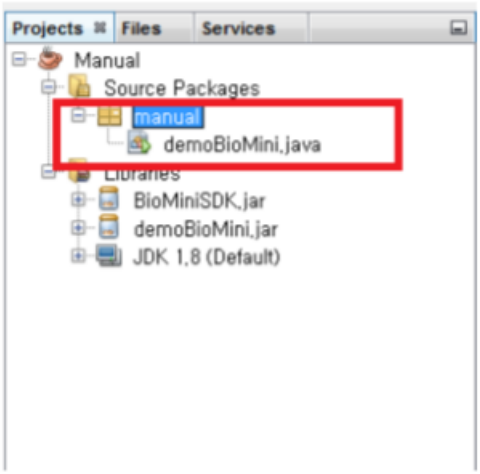
1. **After creating the project, select "Add JAR/Folder…" by using right mouse button from the Library.**



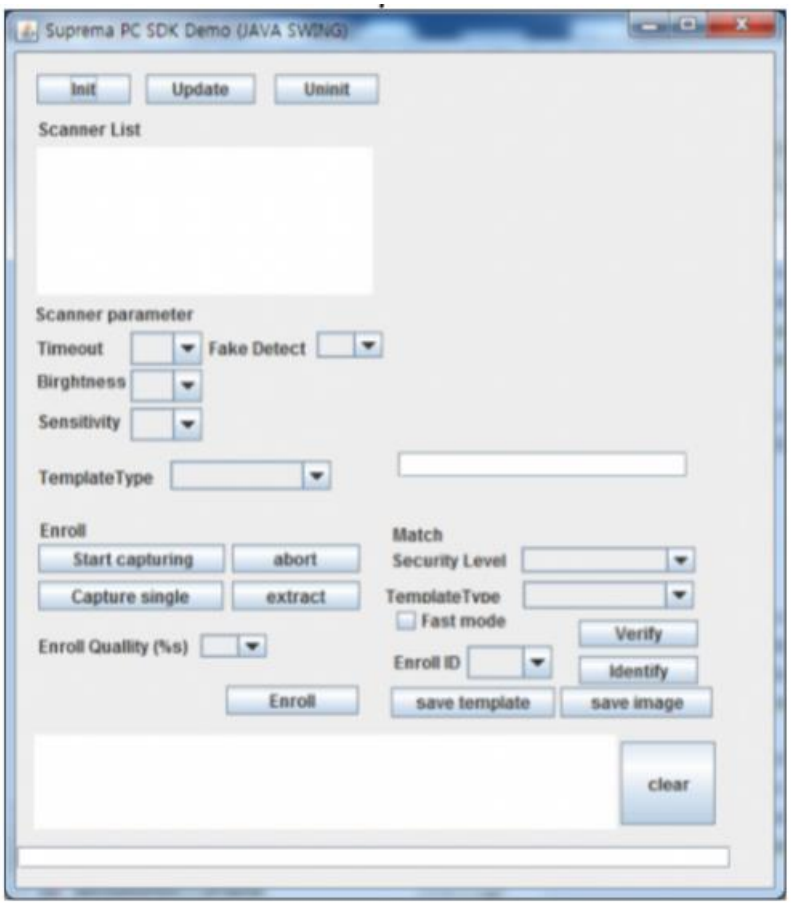
2. **Add BioMiniSDK.jar & demoBioMini.jar from the bin folder as below.**



3. **Add /SDK_Dir/samples/java\demoBioMini.java file to the source packages.**



4. **Can run as below once initiated after compile.**



4.2. API REFERENCE

UFScanner Functions

UFS_Init

Initializes a UFScanner module.

Syntax

```
int UFS_Init();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

```
int nRes =;
BioMiniSDK p = null;
//make instance p = new BioMiniSDK();
nRes = p.UFS_Init();
if (nRes ==p.UFS_OK) {
	// UFS_Init is succeeded
}
else {
	// UFS_Init is failed
	// Use UFS_GetErrorString method to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_Update

Enforces a UFScanner module to update the connection state of scanners.

Syntax

```
int UFS_Update();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;

//make class library instance(BioMiniSDK p)

ufs_res = p.UFS_Update();
if (ufs_res == p.UFS_OK) {
	// UFS_Update is succeeded
}
else {
	// UFS_Update is failed
	// Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_Uninit

Un-initializes a UFScanner module.

Syntax

```
int UFS_UnInit();
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;

//make class library instance(BioMiniSDK p)

ufs_res = p.UFS_Uninit();
if (ufs_res == p.UFS_OK) {
	// UFS_Uninit is succeeded
}
else {
	// UFS_Uninit is failed
	// Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SetScannerCallback

Registers the scanner callback function.

Syntax

```
int UFS_SetScannerCallback(String nCallbackFunctionName);
```

Parameters

- nCallbackFunctionName[in]: Name of the scanner callback

Return value

UFS_STATUS

Remarks

Examples

```
// Define scanner procedure

public void scannerCallback(char[] szScannerID, int bSensorOn)
{
	// ...
}
// Set parameter, the name of call function for scanner event (USB Plug)
//make class library instance(BioMiniSDK p)

ufs_res = p.UFS_SetScannerCallback("scannerCallback");
if (ufs_res==p.UFS_OK) {
	// UFS_SetScannerCallback is succeeded
}
else {
	// UFS_SetScannerCallback is failed
	// Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_RemoveScannerCallback

Removes the registered scanner callback function.

Syntax

```
int UFS_RemoveScannerCallback()
```

Parameters

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;

//make class library instance(BioMiniSDK p)

ufs_res = p.UFS_RemoveScannerCallback();
if (ufs_res == UFS_OK) {
	// UFS_RemoveScannerCallback is succeeded
}
else {
	// UFS_RemoveScannerCallback is failed
	// Use UFS_GetErrorString function to show error string
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerNumber

Gets the number of scanners.

Syntax

```
int UFS_GetScannerNumber(int[] pnScannerNumber);
```

Parameters

- pnScannerNumber [out]: Receive the number of scanners

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
int[] nNumber = new int[1];

//make class library instance(BioMiniSDK p)

ufs_res = p.UFS_GetScannerNumber(nNumber);
if (ufs_res == UFS_OK) {
	// UFS_GetScannerNumber is succeeded
	nNumber[1] ...
}
else {
	// UFS_GetScannerNumber is failed
	// Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerHandle

Gets the scanner handle using a scanner index.

Syntax

```
int UFS_GetScannerHandle(int nScannerIndex,
                         long[] phScanner );
```

Parameters

- nScannerIndex [in]: Scanner index (0 ~ number of scanners - 1)
- phScanner [out]: Pointer to handle of the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
int index = ;

// Set nScannerIndex to (0 ~ number of scanners - 1 )
// Number of scanner can be retrieved using UFS_GetScannerNumber function
//make class library instance(BioMiniSDK p)

ufs_res = p.UFS_GetScannerHandle(index, hScanner);
if (ufs_res == UFS_OK) {
      // UFS_GetScannerHandle is succeeded
      // hScanner[0] is the handle
}
else {
      // UFS_GetScannerHandle is failed
      // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerHandleByID

Gets the scanner handle using a scanner ID.

Syntax

```
int UFS_GetScannerHandleByID(String szScannerID,
                             long[] phScanner );
```

Parameters

- szScannerID [in]: Scanner ID
- phScanner [out]: Pointer to handle of the scanner object

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerIndex

Gets a scanner index that is assigned to the scanner handle.

Syntax

```
int UFS_GetScannerIndex(long hScanner,
                        int[] pnScannerIndex );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pnScannerIndex [out]: Receive scanner index of specified scanner handle

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerID

Gets scanner ID assigned to scanner handle.

Syntax

```
int UFS_GetScannerID(long hScanner,
                     byte[] szScannerID );
```

Parameters

- hScanner [in]: Handle to the scanner object
- szScannerID [out]: *Receive scanner ID of specified scanner handle; Scanner ID has maximum 32 characters. szScannerID must be allocated in user's applications and allocated size must be larger than 33 bytes for considering null character in 33th byte position.*

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
byte[] strID = new byte[128];

// Should be larger than 33 bytes
// make class library instance(BioMiniSDK p)
// Get hScanner handle

ufs_res = p.UFS_GetScannerID(hScanner[], strID);
if (ufs_res == p.UFS_OK) {
      // UFS_GetScannerID is succeeded
      // byte to string..
}
else {
      // UFS_GetScannerID is failed
      // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetScannerType

Gets the scanner type that is assigned to the scanner handle.

Syntax

```
int UFS_GetScannerType(long hScanner,
                       int[] pnScannerType );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pnScannerType [out]: Receives one of the scanner type

Scanner type	Code	Description
UFS_SCANNER_TYPE_SFR200	1001	Suprema SFR200
UFS_SCANNER_TYPE_SFR300	1002	Suprema SFR300-S
UFS_SCANNER_TYPE_SFR300v2	1003	Suprema SFR300v2, SFR400
UFS_SCANNER_TYPE_SFR500	1004	BioMini Plus
UFS_SCANNER_TYPE_SFR600	1005	BioMini Slim
UFS_SCANNER_TYPE_SFR410	1006	BioMini OC4
UFS_SCANNER_TYPE_SFR550	1007	BioMini Plus 2
UFS_SCANNER_TYPE_SFR700	1008	BioMini Slim 2
UFS_SCANNER_TYPE_BMS3	1012	BioMini Slim 3

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
int[] nScannerType = new int[1];

// make class library instance(BioMiniSDK p)

// Get hScanner handle

ufs_res = p.UFS_GetScannerType(hScanner[], nScannerType);
if (ufs_res == UFS_OK) {
	// UFS_GetScannerType is succeeded
	// nScannerType[0] is the scanner type
}
else {
	// UFS_GetScannerType is failed
	// Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetParameter

Gets parameter value of UFScanner module.

Syntax

```
int UFS_GetParameter(long hScanner,
                     int nParam,
                     int[] pValue );
```

Parameters

- hScanner [in]: Handle to the scanner object
- nParam [in]: Parameter type; one of parameters

Parameter	Code	Description	Default value
UFS_PARAM_TIMEOUT	201	Timeout (millisecond unit) (0: infinite)	5000
UFS_PARAM_BRIGHTNESS	202	Brightness (0 ~ 255); Higher value means darker image. * Supported Device: BioMini (SFR400, SFR410) BioMini Plus (SFU-500)	100
UFS_PARAM_SENSITIVITY	203	Sensitivity (0 ~ 7); Higher value means more sensitive	4
UFS_PARAM_SCANNING_MODE	220	Adjust the image size of BioMini Plus 2 (0: 288×340[pixels], 1: 315×354[pixels])	0
UFS_PARAM_DETECT_CORE	301	Detect core (0: not use core, 1: use core)	0
UFS_PARAM_TEMPLATE_SIZE	302	Template size (byte unit) (256 ~ 1024, 32 bytes step size)	1024
UFS_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFS_PARAM_DETECT_FAKE	312	Use live Finger Detection (0: not use LFD, 1 ~3 : use LFD); Higher value means more strong to fake finger * Supported Device: BioMini Slim(SFU-S20)	0
UFS_PARAM_LFD_TYPE	313	LFD operation options (0: UFS_LFD_TYPE_DEFAULT, 1: UFS_LFD_TYPE_ADVANCED; LFD checking level is upgrade so it can be expected improved defense performance againt to some counterfeit fingerprints.) *Supported device : BioMini Slim (SFU-S20)	0
UFS_PARAM_LFD_FILE	314	Specify the path of the engine file for upgrading the LFD Engine. Since the default engine is built into the SDK, you do not need to call it unless it is for upgrade purposes. * Supported device : BioMini Plus 2 (SFR550), BioMini Slim (SFU-S20)	-
UFS_PARAM_LFD_SCORE	317	LFD score * Supported device: BioMini Slim 2, BioMini Slim 3	

UFS_PARAM_LANGUAGE	401	Language selection at runtime of EnrollUI	-

Parameter	Code	Description			Default value
UFS_PARAM_FPQUALITY_MODE	402	Quality score type			0
		UFS_NQS_MODE_DEFAULT	0	Suprema quality score; Quality value ranges from 0(Poorest) to 100(Highest)	
		UFS_NQS_MODE_NFIQ_PERCENTILE	1	NFIQ 1.0 quality score; 5 levels of quality value between 20(Poorest) and 100(Highest)	
		UFS_NQS_MODE_NFIQ	2	NFIQ 1.0 quality score; 5 levels of quality value between 1(Highest) and 5(Poorest)	
		UFS_NQS_MODE_NFIQ2	3	NFIQ 2.0 quality score; Quality value ranges from 0(Poorest) to 100(Highest)	
UFS_PARAM_NFIQ2_FILE	406	Specify the path of the NFIQ 2.0 Engine file - The NFIQ2.0 Library file in the BioMini PC SDK package is “NFIQ2.dll”. (If not set, you cannot use NFIQ2.0.)			N/A

- pValue [out]: Receives parameter value of specified parameter type;
  pValue must point to adequate storage type matched to parameter type

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
int[] nValue = new int[1];

//make class library instance(BioMiniSDK p)

// Get hScanner handle

// Get timeout
ufs_res = p.UFS_GetParameter(hScanner[], p.UFS_PARAM_TIMEOUT, nValue );
// Error handling routine is omitted
// nValue[0] is the parameter value

// Get brightness
ufs_res = p.UFS_GetParameter(hScanner[], p.UFS_PARAM_BRIGHTNESS, nValue );
// Error handling routine is omitted

// Get sensitivity
ufs_res = p.UFS_GetParameter(hScanner[], p.UFS_PARAM_SENSITIVITY, nValue );
// Error handling routine is omitted

// Get detect core
ufs_res = p.UFS_GetParameter(hScanner[], p.UFS_PARAM_DETECT_CORE, nValue );
// Error handling routine is omitted

// Get template size
ufs_res = p.UFS_GetParameter(hScanner[], p.UFS_PARAM_TEMPLATE_SIZE, nValue );
// Error handling routine is omitted

// Get use SIF
ufs_res = p.UFS_GetParameter(hScanner[], p.UFS_PARAM_USE_SIF, nValue );
// Error handling routine is omitted
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SetParameter

Sets parameter value of UFScanner module.

Syntax

```
int UFS_SetParameter(long hScanner,
                     int nParam,
                     int[] pValue )
```

Parameters

- hScanner [in: Handle to the scanner object
- nParam [in]: Parameter type; one of parameters

Parameter	Code	Description	Default value
UFS_PARAM_TIMEOUT	201	Timeout (millisecond unit) (0: infinite)	5000
UFS_PARAM_BRIGHTNESS	202	Brightness (0 ~ 255); Higher value means darker image. * Supported Device: BioMini (SFR400, SFR410) BioMini Plus (SFU-500)	100
UFS_PARAM_SENSITIVITY	203	Sensitivity (0 ~ 7); Higher value means more sensitive	4
UFS_PARAM_SCANNING_MODE	220	Adjust the image size of BioMini Plus 2 (0: 288×340[pixels], 1: 315×354[pixels])	0
UFS_PARAM_DETECT_CORE	301	Detect core (0: not use core, 1: use core)	0
UFS_PARAM_TEMPLATE_SIZE	302	Template size (byte unit) (256 ~ 1024, 32 bytes step size)	1024
UFS_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFS_PARAM_DETECT_FAKE	312	Use live Finger Detection (0: not use LFD, 1 ~3 : use LFD); Higher value means more strong to fake finger * Supported Device: BioMini Slim(SFU-S20)	0
UFS_PARAM_LFD_TYPE	313	LFD operation options (0: UFS_LFD_TYPE_DEFAULT, 1: UFS_ LFD_TYPE_ADVANCED; LFD checking level is upgrade so it can be expected improved defense performance againt to some counterfeit fingerprints.) *Supported device : BioMini Slim (SFU-S20)	0
UFS_PARAM_LFD_FILE	314	Specify the path of the engine file for upgrading the LFD Engine. Since the default engine is built into the SDK, you do not need to call it unless it is for upgrade purposes. * Supported device : BioMini Plus 2 (SFR550), BioMini Slim (SFU-S20)	-
UFS_PARAM_LANGUAGE	401	Language selection at runtime of EnrollUI	-

Parameter	Code	Description	Default

<table>
<tr><th colspan="4"></th><th>value</th></tr>
<tr><td rowspan="5">UFS_PARAM_FPQUALITY_MODE</td><td rowspan="5">402</td><td colspan="3">Quality score type</td><td rowspan="5">0</td></tr>
<tr><td>UFS_NQS_MODE_DEFAULT</td><td>0</td><td>Suprema quality score; Quality value ranges from 0(Poorest) to 100(Highest)</td></tr>
<tr><td>UFS_NQS_MODE_NFIQ_PERCENTILE</td><td>1</td><td>NFIQ 1.0 quality score; 5 levels of quality value between 20(Poorest) and 100(Highest)</td></tr>
<tr><td>UFS_NQS_MODE_NFIQ</td><td>2</td><td>NFIQ 1.0 quality score; 5 levels of quality value between 1(Highest) and 5(Poorest)</td></tr>
<tr><td>UFS_NQS_MODE_NFIQ2</td><td>3</td><td>NFIQ 2.0 quality score; Quality value ranges from 0(Poorest) to 100(Highest)</td></tr>
<tr><td>UFS_PARAM_NFIQ2_FILE</td><td>406</td><td colspan="3">Specify the path of the NFIQ 2.0 Engine file - The NFIQ2.0 Library file in the BioMini PC SDK package is “NFIQ2.dll”. (If not set, you cannot use NFIQ2.0.)</td><td>N/A</td></tr>
</table>

- pValue [in]: Pointer to parameter value of specified parameter type;
  pValue must point to adequate storage type matched to parameter type

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
int[] nValue = new int[1];

//make class library instance(BioMiniSDK p)

// Get hScanner handle

// Set timeout to nValue nValue[0] = 5000;
ufs_res = p.UFS_SetParameter(hScanner[], p.UFS_PARAM_TIMEOUT, nValue);
// Error handling routine is omitted

// Set brightness to nValue nValue[0] = 100;
ufs_res = p.UFS_SetParameter(hScanner[], p.UFS_PARAM_BRIGHTNESS, nValue);
// Error handling routine is omitted

// Set sensitivity to nValue
ufs_res = p.UFS_SetParameter(hScanner[], p.UFS_PARAM_SENSITIVITY, nValue);
// Error handling routine is omitted

// Set detect core to nValue
ufs_res = p.UFS_SetParameter(hScanner[], p.UFS_PARAM_DETECT_CORE, nValue);
// Error handling routine is omitted

// Set template size to nValue
ufs_res = p.UFS_SetParameter(hScanner[], p.UFS_PARAM_TEMPLATE_SIZE, nValue);
// Error handling routine is omitted

// Set use SIF to nValue
ufs_res = p.UFS_SetParameter(hScanner[], p.UFS_PARAM_USE_SIF, nValue);
// Error handling routine is omitted
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_IsSensorOn

Checks whether a scanner is connected or not.

Syntax

```
int UFS_IsSensorOn(long hScanner,
                   int[] pbSensorOn );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pbSensorOn [out]: Receive the status of specified scanner object;
  1: the scanner is connected, 0: the scanner is disconnected

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_IsFingerOn

Checks whether a finger is placed on a scanner or not.

Syntax

```
int UFS_IsFingerOn(long hScanner,
                   int[] pbFingerOn );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pbFingerOn [out]: Checks if a finger is placed on the specified scanner;
  1: a finger is on the scanner, 0: a finger is not on the scanner

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_CaptureSingleImage

Captures single image. Captured image is stored to the internal buffer.

Syntax

```
int UFS_CaptureSingleImage(long hScanner);
```

Parameters

- hScanner [in]: Handle to the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];

// make class library instance(BioMiniSDK p)

// Get hScanner handle

ufs_res = p.UFS_CaptureSingleImage(hScanner[]);
if (ufs_res == p.UFS_OK) {
	// UFS_CaptureSingleImage is succeeded
}
else {
	// UFS_CaptureSingleImage is failed
	// Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_StartCapturing

Starts capturing. The capture is going on until the specified time exceeds.

Syntax

```
int UFS_StartCapturing(long hScanner,
                       String nCallbackFunctionName )
```

Parameters

- hScanner [in]: Handle to the scanner object
- nCallbackFunctionName [in]: Name of the capture callback

Return value

UFS_STATUS

Remarks

Examples

```
// Define capture procedure

public void captureCallback(int bFingerOn, byte[] pImage, int nWidth, int nHeight, int nResolution)
{
    // ....
    // pMainInstance.drawCurrentFingerImage();
}

int ufs_res;
long[] hScanner = new long[1];

// make class library instance(BioMiniSDK p)

// Get hScanner handle

// Set parameter, the name of your call function for getting captured image

ufs_res = p.UFS_StartCapturing(hScanner[], "captureCallback");

if (ufs_res == p.UFS_OK) {
    // UFS_StartCapturing is succeeded
}
else {
    // UFS_StartCapturing is failed
    // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_StartAutoCapture

Starts the automatic capture. Currently this function is working for Suprema SFR600(BioMini Slim) and SFR700(BioMini Slim 2).

Syntax

```
int UFS_StartAutoCapture(long hScanner,
                         String nCallbackFunctionName )
```

Parameters

- hScanner [in]: Handle to the scanner object
- nCallbackFunctionName [in]: Name of the capture callback

Return value

UFS_STATUS

Remarks

Examples

```
// Define capture procedure

public void captureCallback(int bFingerOn, byte[] pImage, int nWidth, int nHeight, int nResolution)
{
    // ....
    // pMainInstance.drawCurrentFingerImage();
}

int ufs_res;
long[] hScanner = new long[1];

// make class library instance(BioMiniSDK p)

// Get hScanner handle

// Set parameter, the name of your call function for getting captured image

ufs_res = p.UFS_StartAutoCapture(hScanner[], "captureCallback");
if (ufs_res == p.UFS_OK) {
    // UFS_StartAutoCapture is succeeded
}
else {
    // UFS_StartAutoCapture is failed
    // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_IsCapturing

Checks if the specified scanner is running to capture which is started by UFS_CaptureSingleImage or UFS_StartCapturing

Syntax

```
int UFS_IsCapturing(long hScanner,
                    int[] bCapturing );
```

Parameters

- hScanner [in]: Handle to the scanner object
- bCapturing [out]: Checks if the specified scanner is running capturing;
  1: the capture is running, 0: the capture is not running

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_STATUS

UFS_AbortCapturing

Aborts capturing which is started by UFS_CaptureSingleImage or UFS_StartCapturing.

Syntax

```
int UFS_AbortCapturing(long hScanner);
```

Parameters

- hScanner [in]: Handle to the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];

// make class library instance(BioMiniSDK p)

// Get hScanner handle

ufs_res = p.UFS_AbortCapturing(hScanner[]);
if (ufs_res == p.UFS_OK) {
	// UFS_AbortCapturing is succeeded
}
else {
	// UFS_AbortCapturing is failed
	// Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_STATUS

UFS_Extract

Extracts a template from the stored image buffer which is acquired using UFS_CaptureSingleImage or UFS_StartCapturing.

Syntax

```
int UFS_Extract(long hScanner,
                byte[] pTemplate,
                int[] pnTemplateSize,
                int[] pnEnrollQuality );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplate [out]: Pointer to the template array; The array must be allocated in advance
- pnTemplateSize [out]: Receives the size (in bytes) of pTemplate
- pnEnrollQuality [out]: Receives the quality of enrollment; Quality value ranges from 1 to 100. Typically, this value should be above 30 for further processing such as enroll and matching. Especially in case of enrollment, the use of good quality image (above 50) is highly recommended.

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_STATUS

UFS_ExtractEx

Extracts a template from the stored image buffer which is acquired using UFS_CaptureSingleImage or UFS_StartCapturing. This is extended version of UFS_Extract function to accommodate large size template.

Syntax

```
int UFS_ExtractEx(long hScanner,
                  int nBufferSize,
                  byte[] pTemplate,
                  int[] pnTemplateSize,
                  int[] pnEnrollQuality );
```

Parameters

- hScanner [in]: Handle to the scanner object
- nBufferSize [in] : Template buffer size
- pTemplate [out] : Pointer to the template array; The array must be allocated in advance
- pnTemplateSize [out] : Receives the size (in bytes) of pTemplate
- pnEnrollQuality [out] : Receives the quality of enrollment; Quality value ranges from 1 to 100. Typically, this value should be above 30 for further processing such as enroll and matching. Especially in case of enrollment, the use of good quality image (above 50) is highly recommended.

Return value

UFS_STATUS

Remarks

Examples

```
// Template size can be controlled by using UFS_SetParameter function
// Default value is 1024 bytes
int MAX_TEMPLATE_SIZE = 1024;

int ufs_res;
long[] hScanner = new long[1];
byte[] Template = new byte[MAX_TEMPLATE_SIZE];
int[] TemplateSize= new int[1];
int[] TemplateQuality= new int[1];

// make class library instance(BioMiniSDK p)

// Get hScanner handle

ufs_res = p.UFS_ExtractEx(hScanner[], MAX_TEMPLATE_SIZE, Template,
                          TemplateSize, TemplateQuality);
if (ufs_res == p.UFS_OK) {
      // UFS_ExtractEx is succeeded
}
else {
      // UFS_ExtractEx is failed
      // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_CaptureSingleImage
UFS_StartCapturing
UFS_Extract
UFS_STATUS

UFS_SetEncryptionKey

Sets encryption key.

Syntax

```
int UFS_SetEncryptionKey(long hScanner,
                         byte[] pKey );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pKey[out]: Pointer to the 32 bytes key array;
  default key is first byte is 1 and second to 32th byte are all

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_EncryptTemplate

Encrypts template.

Syntax

```
int UFS_EncryptTemplate(long hScanner,
                        byte[] pTemplateInput,
                        int nTemplateInputSize,
                        byte[] pTemplateOutput,
                        int[] pnTemplateOutputSize );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplateInput [in]: Pointer to input template data
- nTemplateInputSize [in]: Input template size
- pTemplateOutput [out]: Pointer to encrypted template data
- pnTemplateOutputSize [in / out]: Inputs allocated size of encrypted template data; Receives output template size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_DecryptTemplate

Decrypts template.

Syntax

```
int UFS_DecryptTemplate(long hScanner,
                        byte[] pTemplateInput,
                        int nTemplateInputSize,
                        byte[] pTemplateOutput,
                        int[] pnTemplateOutputSize )
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplateInput [in]: Pointer to input template data(encrypted)
- nTemplateInputSize [in]: Input template size
- pTemplateOutput [out]: Pointer to output template data
- pnTemplateOutputSize [in / out]: Inputs allocated size of output template data; Receives output template size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferInfo

Gets the information of the capture image buffer.

Syntax

```
int UFS_GetCaptureImageBufferInfo(long hScanner,
                                  int[] pnWidth,
                                  int[] pnHeight,
                                  int[] pnResolution );
```

Parameters

- Parameters hScanner [in]: Handle to the scanner object
- pnWidth [out]: Receives the width of the capture image buffer
- pnHeight [out]: Receives the height of the capture image buffer
- pnResolution [out]: Receives the resolution of the capture image buffer

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
int[] nWidth = new int[1];
int[] nHeight = new int[1];
int[] nResolution = new int[1];

// make class library instance(BioMiniSDK p)

// Get hScanner handle

ufs_res = p.UFS_GetCaptureImageBufferInfo(hScanner[], nWidth, nHeight, nResolution);
if (ufs_res == p.UFS_OK) {
      // UFS_GetCaptureImageBufferInfo is succeeded
}
else {
      // UFS_GetCaptureImageBufferInfo is failed
      // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBuffer

Copies the capture image buffer to the specified image data array.

Syntax

```
int UFS_GetCaptureImageBuffer(long hScanner,
                              byte[] pImageData )
```

Parameters

- hScanner [in]: Handle to the scanner object
- pImageData [out]: Pointer to image data array;
  The array must be allocated bigger than the size of capture image buffer in advance

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
int[]nWidth = new int[1];
int[]nHeight= new int[1];
int[]nResolution= new int[1];

// Get hScanner handle

// Get capture image buffer information
ufs_res = p.UFS_GetCaptureImageBufferInfo(hScanner[], nWidth, nHeight, nResolution);
// Error handling routine is omitted

// Allocate image buffer
byte[] pImageData = new byte[nWidth * nHeight];

ufs_res = p.UFS_GetCaptureImageBuffer(hScanner[], pImageData);
if (ufs_res == p.UFS_OK) {
      // UFS_GetCaptureImageBuffer is succeeded
}
else {
      // UFS_GetCaptureImageBuffer is failed
      // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferToBMPImageBuffer

Copies the capture image buffer to the specified image data of bmp format.

Syntax

```
int UFS_GetCaptureImageBufferToBMPImageBuffer(long hScanner,
                                              byte[] pImageData,
                                              int[] pImageLength );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pImageData [out]: Pointer to bmp image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance
- pImageLength [out]: pointer to bmp image data size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferTo19794_4ImageBuffer

Copies the capture image buffer to the specified image data of 19794_4 format.

Syntax

```
int UFS_GetCaptureImageBufferTo19794_4ImageBuffer(long hScanner,
                                                  byte[] pImageData,
                                                  int[] pImageLength );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pImageData [out]: Pointer to 19794_4 format image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance
- pImageLength [out]: pointer to 19794_4 format image data size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferToWSQImageBuffer

Copies the capture image buffer to the specified image data of the WSQ format.

Syntax

```
int UFS_GetCaptureImageBufferToWSQImageBuffer(long hScanner,
                                              float ratio,
                                              byte[] wsqData,
                                              int[] wsqDataLen );
```

Parameters

- hScanner [in]: Handle to the scanner object
- ratio [in]: Compression ratioof image
- wsqData [out]: Pointer to WSQ format image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance
- wsqDataLen [out]: pointer to WSQ format image data size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetCaptureImageBufferToWSQImageBufferVar

Copies the capture image buffer
(cropped or expanded to the specified size) to the target image data buffer of the WSQ format.

Syntax

```
int UFS_GetCaptureImageBufferToWSQImageBufferVar(long hScanner,
                                                 float ratio,
                                                 byte[] wsqData,
                                                 int[] wsqDataLen,
                                                 int nWidth,
                                                 int nHeight );
```

Parameters

- hScanner [in]: Handle to the scanner object
- ratio [in]: Compression ratio of image
- wsqData [out]: Pointer to WSQ format image data;
  The buffer must be allocated bigger than the size of capture image buffer in advance
- wsqDataLen [out]: pointer to WSQ format image data size
- nWidth [in]: Width to resize the capture image
- nHeight [in]: Height to resize the capture image

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_DecompressWSQBMP

Decompress WSQ file and save to BMP file.

Syntax

```
int UFS_ DecompressWSQBMP(long hScanner,
                          String wsqFile,
                          String bmpFile );
```

Parameters

- hScanner [in]: Handle to the scanner object
- wsqFile [in]: Specifies file name to get wsq data buffer
- bmpFile [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_DecompressWSQBMPMem

Decompress WSQ buffer and save to image data of bmp format.

Syntax

```
int UFS_ DecompressWSQBMPMem(long hScanner,
                             byte[] wsqBuffer,
                             int wsqBufferLen,
                             byte[] bmpBuffer,
                             int[] bmpBufferLen );
```

Parameters

- hScanner [in]: Handle to the scanner object
- wsqBuffer [in]: Pointer to WSQ format image data
- wsqBufferLen [in]: Size of WSQ format image data
- bmpBuffer [out]: Pointer to bmp image data;
  The array must be allocated bigger than the size of capture image buffer in advance.
- bmpBufferLen [out]: pointer to bmp image data size

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SaveCaptureImageBufferToBMP

Saves the capture image buffer to the specified file of the bitmap format.

Syntax

```
int UFS_SaveCaptureImageBufferToBMP(long hScanner,
                                    String szFileName );
```

Parameters

- hScanner [in]: Handle to the scanner object
- szFileName [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
String szFileName;

// make class library instance(BioMiniSDK p)

// Get hScanner handle

// Get file name, szFileName

ufs_res = p.UFS_SaveCaptureImageBufferToBMP(hScanner[], szFileName);
if (ufs_res == p.UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
}
else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SaveCaptureImageBufferTo19794_4

Saves the capture image buffer to the specified file of the 19794_4 format.

Syntax

```
java int UFS_SaveCaptureImageBufferTo19794_4(long hScanner,
                                             String szFileName )
```

Parameters

- hScanner [in]: Handle to the scanner object
- szFileName [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
String szFileName;

// make class library instance(BioMiniSDK p)

// Get hScanner handle

// Get file name, szFileName

ufs_res = p.UFS_SaveCaptureImageBufferTo19794_4(hScanner[], szFileName);
if (ufs_res == p.UFS_OK) {
      // UFS_SaveCaptureImageBufferTo19794_4 is succeeded
}
else {
      // UFS_SaveCaptureImageBufferTo19794_4 is failed
      // Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SaveCaptureImageBufferToWSQ

Saves the capture image buffer to the specified file of the WSQ format.

Syntax

```
int UFS_SaveCaptureImageBufferToWSQ(long hScanner,
                                    float ratio,
                                    String szFileName);
```

Parameters

- hScanner [in]: Handle to the scanner object
- ratio [in]: Compression ratio of image
- szFileName [in]: Specifies file name to save image buffer

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SaveCaptureImageBufferToWSQVar

Saves the capture image buffer (cropped or expanded to the specified size) to the target file of the WSQ format.

Syntax

```
int UFS_SaveCaptureImageBufferToWSQVar(long hScanner,
                                       float ratio,
                                       String szFileName,
                                       int nWidth,
                                       int nHeight);
```

Parameters

- hScanner [in]: Handle to the scanner object
- ratio [in]: Compression ratio of image
- szFileName [in]: Specifies file name to save image buffer
- nWidth [in]: Width to resize the capture image
- nHeight [in]: Height to resize the capture image

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_ClearCaptureImageBuffer

Clears the capture image buffer stored to the internal buffer.

Syntax

```
int UFS_ClearCaptureImageBuffer(long hScanner);
```

Parameters

- hScanner [in]: Handle to the scanner object

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];

// make class library instance(BioMiniSDK p)

// Get hScanner handle

ufs_res = p.UFS_ClearCaptureImageBuffer(hScanner[]);
if (ufs_res == p.UFS_OK) {
	// UFS_ClearCaptureImageBuffer is succeeded
}
else {
	// UFS_ClearCaptureImageBuffer is failed
	// Use UFS_GetErrorString function to show error string
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetErrorString

Gets the error string for specified UFS_STAUS value.

Syntax

```
int UFS_GetErrorString(int res,
                       byte[] szErrorString );
```

Parameters

- hScanner [in]: Handle to the scanner object
- szErrorString [out]: Receives error string

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
byte[] strError = new byte[128];

// Get status return code, ufs_res

ufs_res = p.UFS_GetErrorString(ufs_res, strError);
if (ufs_res == p.UFS_OK) {
    // UFS_GetErrorString is succeeded
}
else {
    // UFS_GetErrorString is failed
}
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetTemplateType

Gets the template type value.

Syntax

```
int UFS_GetTemplateType(long hScanner,
                        int[] nTemplateType );
```

Parameters

- hScanner [in]: Handle to the scanner object
- nTemplateType [out]: Receives the parameter value of specified parameter type; 'pValue' must point to adequate type that is matched with the parameter type

Template type	Code	Description
UFS_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFS_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFS_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SetTemplateType

Sets the template type value.

Syntax

```
int UFS_SetTemplateType(long hScanner,
                        int nTemplateType );
```

Parameters

- hScanner [in]: Handle to the scanner object
- nTemplateType [in]: Parameter type; one of template type

Template type	Code	Description
UFS_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFS_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFS_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFS_STATUS

Remarks

Examples

```
int ufs_res;
long[] hScanner = new long[1];
int nTemplateType;

// make class library instance(BioMiniSDK p)

// Get hScanner handle

nTemplateType =UFS_TEMPLATE_TYPE_SUPREMA;

ufs_res = p.UFS_SetTemplateType(hScanner[], nTemplateType);
// Error handling routine is omitted
```

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SelectTemplate

Selects n number of good templates from m number of input templates.

Syntax

```
int UFS_SelectTemplate(long hScanner,
                       byte[][] ppTemplateInput,
                       int[] pnTemplateInputSize,
                       int nTemplateInputNum,
                       byte[][] ppTemplateOutput,
                       int[] pnTemplateOutputSize,
                       int nTemplateOutputNum );
```

Parameters

- hScanner [in]: Handle to the scanner object
- ppTemplateInput [in]: Array pointer to the input template arrays
- pnTemplateInputSize [in]: Array pointer to input templates' size
- nTemplateInputNum [in]: Number of input templates
- ppTemplateOutput [out]: Array pointer to the output template arrays
- pnTemplateOutputSize [out]: Array pointer to the output templates' size
- nTemplateOutputNum [in]: Number of output templates; should be less than input template number by more than one

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_SelectTemplateEx

Selects n number of good templates from m number of input templates. This is extended version of UFS_SelectTemplate function to accommodate large size template.

Syntax

```
int UFS_SelectTemplateEx(long hScanner,
                         int nBufferSize,
                         byte[][] ppTemplateInput,
                         int[] pnTemplateInputSize,
                         int nTemplateInputNum,
                         byte[][] ppTemplateOutput,
                         int[] pnTemplateOutputSize,
                         int nTemplateOutputNum );
```

Parameters

- hScanner [in]: Handle to the scanner object
- nBufferSize [in]: Template buffer size
- ppTemplateInput [in]: Array pointer to the input template arrays
- pnTemplateInputSize [in]: Array pointer to the input templates' size
- nTemplateInputNum [in]: Number of input templates
- ppTemplateOutput [out]: Array pointer to the output template arrays
- pnTemplateOutputSize [out]: Array pointer to the output templates' size
- nTemplateOutputNum [in]: Number of output templates; should be less than input template number by more than one

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_SelectTemplate
UFS_STATUS

UFS_GetFPQuality

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

Syntax

```
int UFS_GetFPQuality(long hScanner,
                     byte[] pFPImage,
                     int nWidth,
                     int nHeight,
                     int[] pnFPQuality );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pFPImage [in]: Raw capture image data
- nWidth [in]: Width of capture image data
- nHeight [in]: Height of capture image data
- pnFPQuality [in]: NIST quality score of image data

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_GetFeatureNumber

Get the number of Minutiae from the template data.

Syntax

```
int UFS_GetFeatureNumber(long hScanner,
                         byte[] pTemplate,
                         int nTemplateSize,
                         int[] pnFeatureNum );
```

Parameters

- hScanner [in]: Handle to the scanner object
- pTemplate [in]: Template data
- nTemplateSize [in]: Size of template data
- pnFeatureNum [out]: The number of minutiae from pTemplate

Return value

UFS_STATUS

Remarks

Examples

Requirements

Header	UFScanner.dll
Library	UFScanner.lib

See also

UFS_STATUS

UFS_STATUS

Every function in a UFScanner module returns OK when it succeeds. When it fails, it returns a value corresponding to an error code. Please find the error code on the followings if you'd like to know the information about the UFS_STATUS (Enum) value.

Code	Value	Description
UFS_OK	0	Success
UFS_ERROR	-1	General error
UFS_ERR_DEVICE_NOT_RESPOND	-10	Device is not responded
UFS_ERR_CAPTURE_TIMEOUT	-11	The time for capturing exceeds
UFS_ERR_USB_TIMEOUT	-12	The time for USB transfer exceeds
UFS_ERR_NO_LICENSE	-101	Device is not connected or License is not located
UFS_ERR_LICENSE_NOT_MATCH	-102	License does not match
UFS_ERR_LICENSE_EXPIRED	-103	License has expired
UFS_ERR_NOT_SUPPORTED	-111	This function is not supported
UFS_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFS_ERR_UNKNOWN_TEMPLATE_FORMAT	-120	Abnormal template recognized
UFS_ERR_INVALIDE_ENCRYPTION	-121	Template buffer, encrypted template size is incorrect
UFS_ERR_INVALID_MODE	-122	Device is sleep mode
UFS_ERR_ALREADY_INITIALIZED	-201	Module is already initialized
UFS_ERR_NOT_INITIALIZED	-202	Module is not initialized
UFS_ERR_DEVICE_NUMBER_EXCEED	-203	Device number exceeds
UFS_ERR_LOAD_SCANNER_LIBRARY	-204	Error on loading the library of a scanner
UFS_ERR_CAPTURE_RUNNING	-211	Capturing is started using UFS_CaptureSingleImage of UFS_StartCapturing
UFS_ERR_CAPTURE_FAILED	-212	Capturing is timeout or aborted
UFS_ERR_FAKE_FINGER	-221	Fake finger is detected
UFS_ERR_FINGER_ON_SENSOR	-231	Remove finger from sensor
UFS_ERR_TIMEOUT	-241	Time out
UFS_ERR_NOT_GOOD_IMAGE	-301	Input image is not good
UFS_ERR_EXTRACTION_FAILED	-302	Extraction is failed
UFS_ERR_ID_NOT_FOUND	-500	ID not found on device
UFS_ERR_NO_MATCH_FOUND	-501	Not registered fingerprint
UFS_ERR_NOT_IDENTICAL	-502	Not matched
UFS_ERR_CANNOT_ENROLL	-510	Cannot enroll fingerprint
UFS_ERR_NO_DATA	-511	No data found on device
UFS_ERR_NOT_AUTHORIZED	-520	Not authorized to use this API

In case UFS_PARAM_DETECT_CORE is "1" (=use), you may get the following error

Code	Value	Description
UFS_ERR_CORE_NOT_DETECTED	-351	Core is not detected
UFS_ERR_CORE_TO_LEFT	-352	Move finger to left
UFS_ERR_CORE_TO_LEFT_TOP	-353	Move finger to left-top
UFS_ERR_CORE_TO_TOP	-354	Move finger to top
UFS_ERR_CORE_TO_RIGHT_TOP	-355	Move finger to right-top
UFS_ERR_CORE_TO_RIGHT	-356	Move finger to right
UFS_ERR_CORE_TO_RIGHT_BOTTOM	-357	Move finger to right-bottom
UFS_ERR_CORE_TO_BOTTOM	-358	Move finger to bottom
UFS_ERR_CORE_TO_LEFT_BOTTOM	-359	Move finger to left-bottom
UFS_ERR_FINGER_TOO_RIGHT	-401	Finger is too close to the right edge
UFS_ERR_FINGER_TOO_LEFT	-402	Finger is too close to the left edge
UFS_ERR_FINGER_TOO_TOP	-403	Finger is too close to the top
UFS_ERR_FINGER_TOO_BOTTOM	-404	Finger is too close to the bottom
UFS_ERR_FINGER_TIP	-405	Do not put the fingertip

UFMatcher Functions

UFM_Create

Creates a matcher object.

Syntax

```
int UFM_Create(long hMatcher);
```

Parameters

- hMatcher [out]: Pointer to handle of the matcher object

Return value

UFM_STATUS

Remarks

Examples

```
int ufm_res;
long[] hMatcher = new long[1];

//make instance p = new BioMiniSDK();

ufm_res = p.UFM_Create(hMatcher);
if (ufm_res == p.UFM_OK) {
	// UFM_Create is succeeded
	// hMatcher[0] is the handle
}
else {
	// UFM_Create is failed
	// Use UFM_GetErrorString function to show error string
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_Delete

Deletes specified matcher object.

Syntax

```
int UFM_Delete(long hMatcher);
```

Parameters

- hMatcher [in]: Handle to the matcher object

Return value

UFM_STATUS

Remarks

Examples

```
int ufm_res;
long[] hMatcher = new long[1];

// make class library instance(BioMiniSDK p)

// Create hMatcher handle
ufm_res = p.UFM_Delete(hMatcher[]);
if (ufm_res == p.UFM_OK) {
	// UFM_Delete is succeeded
}
else {
	// UFM_Delete is failed
	// Use UFM_GetErrorString function to show error string
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_GetParameter

Gets parameter value of UFMatcher module.

Syntax

```
int UFM_GetParameter(long hMatcher,
                     int nParam,
                     int[] pValue );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- nParam [in]: Parameter type; one of parameters

Parameter	Code	Description	Default value
UFM_PARAM_FAST_MODE	301	Fast Mode (0: not use fast mode, 1: use fast mode)	1
UFM_PARAM_SECURITY_LEVEL	302	Set the False Accept Ratio(FAR) (1: Below 1%, 2: Below 0.1%, 3: Below 0.01%, 4: Below 0.001%, 5: Below 0.0001%, 6: Below 0.00001%, 7: Below 0.000001%)	4
UFM_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFM_PARAM_AUTO_ROTATE	321	Rotate Mode(0: not use rotate mode, 1: use rotate mode)	0
UFM_PARAM_SDK_VERSION	210	SDK Version (get only)	-
UFM_PARAM_SDK_COPYRIGHT	211	SDK Copyright (get only)	

Return value

UFM_STATUS

Remarks

Examples

```
int ufm_res;
long[] hMatcher = new long[1];
int[] nValue = new int[1];

// make class library instance(BioMiniSDK p)

// Create hMatcher handle

// Get fast mode
ufm_res = p.UFM_GetParameter(hMatcher[], p.UFM_PARAM_FAST_MODE, nValue);
// Error handling routine is omitted

// Get security level
ufm_res = p.UFM_GetParameter(hMatcher[], p.UFM_PARAM_SECURITY_LEVEL, nValue);
// Error handling routine is omitted

// Get use SIF
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_SetParameter

Sets parameter value of UFMatcher module.

Syntax

```
int UFM_SetParameter(long hMatcher,
                     int nParam,
                     int[] pValue );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- nParam [in]: Parameter type; one of parameters

Parameter	Code	Description	Default value
UFM_PARAM_FAST_MODE	301	Fast Mode (0: not use fast mode, 1: use fast mode)	1
UFM_PARAM_SECURITY_LEVEL	302	Set the False Accept Ratio(FAR) (1: Below 1%, 2: Below 0.1%, 3: Below 0.01%, 4: Below 0.001%, 5: Below 0.0001%, 6: Below 0.00001%, 7: Below 0.000001%)	4
UFM_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFM_PARAM_AUTO_ROTATE	321	Rotate Mode(0: not use rotate mode, 1: use rotate mode)	0

Return value

UFM_STATUS

Remarks

Examples

```
int_STATUS ufm_res;
long[] hMatcher = new long[1];
int[] nValue = new int[1];

// make class library instance(BioMiniSDK p)

// Create hMatcher handle

// Set fast mode to nValue
nValue[] = 1;
ufm_res = p.UFM_SetParameter(hMatcher[], p.UFM_PARAM_FAST_MODE, nValue);
// Error handling routine is omitted

// Set security level to nValue
nValue[] = 4;
ufm_res = UFM_SetParameter(hMatcher[], p.UFM_PARAM_SECURITY_LEVEL, nValue);
// Error handling routine is omitted

// Set use SIF to nValue
nValue[] = ;
ufm_res = UFM_SetParameter(hMatcher[], p.UFM_PARAM_USE_SIF, nValue);
// Error handling routine is omitted
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_Verify

Compares two extracted templates.

Syntax

```
int UFM_Verify(long hMatcher,
                byte[] pTemplate1,
                int nTemplate1Size,
                byte[] pTemplate2,
                int nTemplate2Size,
                int[] bVerifySucceed );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- pTemplate1 [in]: Pointer to the template1
- nTemplate1Size [in]: Specifies the size of the template1
- pTemplate2 [in]: Pointer to the template2
- nTemplate2Size [in]: Specifies the size of the template2
- bVerifySucceed [out]: Receives, whether verification is succeed;
  1: verification is succeed, 0: verification is failed

Return value

UFM_STATUS

Remarks

Examples

```
// Assume template size is 1024 bytes int MAX_TEMPLATE_SIZE = 1024;

int ufm_res;
Pointer hMatcher;
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int nTemplate1Size;
int nTemplate2Size;
int[] bVerifySucceed = new int[1];
int nSucceed;

// make class library instance(BioMiniSDK p)

// Create hMatcher handle

// Get two templates, Template1 and Template2

ufm_res = p.UFM_Verify(hMatcher[], Template1, nTemplate1Size, Template2,
nTemplate2Size, bVerifySucceed);
if (ufm_res == p.UFM_OK) {
	// UFM_Verify is succeeded
	nSucceed=bVerifySucceed[];
	if (nSucceed) {
		// Template1 is matched to Template2
	}
	else {
		// Template1 is not matched to Template2
	}
}
else {
	// UFM_Verify is failed
	// Use UFM_GetErrorString function to show error string
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_VerifyEx

Performs same as UFM_Verify, and returns matching score by 6th parameter
(matching score in between 0~1, idle match as the score is close to 1)

Syntax

```
int UFM_Verify(long hMatcher,
                byte[] pTemplate1,
                int nTemplate1Size,
                byte[] pTemplate2,
                int nTemplate2Size,
                float[] fScore,
                int[] bVerifySucceed );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- pTemplate1 [in]: Pointer to the pTemplate1
- nTemplate1Size [in]: Specifies the size of the pTemplate1
- pTemplate2 [in]: Pointer to the pTemplate2
- nTemplate2Size [in]: Specifies the size of the pTemplate2
- fScore [out]: Matching score between pTemplate1 and pTemplate2
- bVerifySucceed [out]: Receives, whether verification is succeed;
  1: verification is succeed, 0: verification is failed

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_Identify, UFM_IdentifyMT

Compares a template with given template array.
UFM_IdentifyMT function uses multi threads internally for faster identifying in multi-core systems.

Syntax

```
int UFM_Identify(long hMatcher,
                 byte[] pTemplate1,
                 int nTemplate1Size,
                 byte[][] pTemplate2,
                 int[] nTemplate2Size,
                 int nTemplate2Num,
                 int nTimeout,
                 int[] nMatchTemplate2Index );
int UFM_IdentifyMT(long hMatcher,
                   byte[] pTemplate1,
                   int nTemplate1Size,
                   byte[][] pTemplate2,
                   int[] nTemplate2Size,
                   int nTemplate2Num,
                   int nTimeout,
                   int[] nMatchTemplate2Index );
```

Parameters

- hMatcher [in]: Handle of the matcher object
- pTemplate1 [in]: Pointer to the template
- nTemplate1Size [in]: Specifies the size of the template
- ppTemplate2 [in]: Pointer to the template array
- pnTemplate2Size [in]: Pointer to the template size array
- nTemplate2Num [in]: Specifies the number of templates in the template array
- nTimeout [in]: Specifies maximum time for identifying in milliseconds; If elapsed time for identifying exceeds nTimeout, function stops further identifying and returns UFM_ERR_MATCH_TIMEOUT;
  0 means infinity
- pnMatchTemplate2Index [out]: Receives the index of matched template in the template array;
  1 means pTemplate1 is not matched to all of templates in ppTemplate2

Return value

UFM_STATUS

Remarks

Examples

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_AbortIdentify

Aborts current identifying procedure started using UFM_Identify.

Syntax

```
int UFM_AbortIdentify(long hMatcher);
```

Parameters

- hMatcher [in]: Handle to the matcher object

Return value

UFM_STATUS

Remarks

Examples

```
int ufm_res;
long[] hMatcher = new long[1];

// make class library instance(BioMiniSDK p)

// Create hMatcher handle

// Start UFM_Identify

ufm_res = p.UFM_AbortIdentify(hMatcher);
if (ufm_res == p.UFM_OK) {
	// UFM_AbortIdentify is succeeded
}
else {
	// UFM_AbortIdentify is failed
	// Use UFM_GetErrorString function to show error string
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_IdentifyInit

Initializes identify with input template.

Syntax

```
int UFM_IdentifyInit(long hMatcher,
                     byte[] pTemplate1,
                     int nTemplate1Size );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- pTemplate1 [in]: Pointer to the template
- nTemplate1Size [in]: Specifies the size of the template

Return value

UFM_STATUS

Remarks

Examples

```
// Assume template size is 1024 bytes int MAX_TEMPLATE_SIZE = 1024;

int ufm_res;
long[] hMatcher = new long[1];
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
int nTemplate1Size;

// make class library instance(BioMiniSDK p)

// Create hMatcher handle

// Get Template1

ufm_res = p.UFM_IdentifyInit(hMatcher[], Template1, nTemplate1Size);
if (ufm_res == p.UFM_OK) {
      // UFM_IdentifyInit is succeeded
}
else {
      // UFM_IdentifyInit is failed
      // Use UFM_GetErrorString function to show error string
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_IdentifyNext

Matches one input template to the template specified in UFM_IdentifyInit.

Syntax

```
int UFM_IdentifyNext(long hMatcher,
                     byte[] pTemplate2,
                     int nTemplate2Size,
                     int[] bIdentifySucceed );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- pTemplate2 [in]: Pointer to the template array
- nTemplate2Size [in]: Specifies the size of the template array
- bIdentifySucceed [out]: Receives whether identification is succeed;
  1: identification is succeed, 0: identification is failed

Return value

UFM_STATUS

Remarks

Examples

```
int MAX_TEMPLATE_NUM = 50;

int ufm_res;
long[] hMatcher = new long[1];
byte[] Template2 = new byte[MAX_TEMPLATE_NUM];
int[] nTemplate2Size = new byte[MAX_TEMPLATE_NUM];
int nTemplate2Num;
int[] bIdentifySucceed = new int[1];
int i;

// make class library instance(BioMiniSDK p)

// Create hMatcher handle

// Get number of templates in DB or something, and save it to nTemplate2Num
```

```
for (i = ; i < nTemplate2Num; i++) {
	// Get one template in DB or something, and save it to Template2 and nTemplate2Size
	ufm_res = p.UFM_IdentifyNext(hMatcher[], Template2[i], nTemplate2Size[i],
	bIdentifySucceed[]);

	if (ufm_res == p.UFM_OK) {
		// UFM_IdentifyNext is succeeded
	}
	else {
		// UFM_IdentifyNext is failed
		// Use UFM_GetErrorString function to show error string
		// return;
	}

	if (bIdentifySucceed[]) {
		// Identification is succeed
		break;
	}
}
if (!bIdentifySucceed[]) {
	// Identification is failed
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_IdentifyInit
UFM_STATUS

UFM_RotateTemplate

Rotates the specified template to the amount of 180 degrees.

Syntax

```
int UFM_RotateTemplate(long hMatcher,
                       byte[] pTemplate,
                       int nTemplateSize );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- pTemplate [in / out]: Pointer to the template
- nTemplateSize [in]: Specifies the size of the template

Return value

UFM_STATUS

Remarks

Examples

```
// Assume template size is 1024 bytes int MAX_TEMPLATE_SIZE = 1024;

int ufm_res;
long[] hMatcher = new long[1];
byte[] Template = new byte[MAX_TEMPLATE_SIZE];
int nTemplateSize;

// make class library instance(BioMiniSDK p)

// Create hMatcher handle

// Get a template, and save it to Template and nTemplateSize

ufm_res = p.UFM_RotateTemplate(hMatcher[], Template, nTemplateSize);
if (ufm_res == p.UFM_OK) {
      // UFM_RotateTemplate is succeeded
}
else {
      // UFM_RotateTemplate is failed
      // Use UFM_GetErrorString function to show error string
}
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_GetErrorString

Gets the error string for specified UFM_STATUS value.

Syntax

```
int UFM_GetErrorString(int res,
                       byte[] szErrorString );
```

Parameters

- res [in]: Status return value
- szErrorString [out]: Receives error string

Return value

UFM_STATUS

Remarks

Examples

```
int ufs_res;
byte[] strError = new byte[128];

// Get status return code, ufm_res
ufs_res = p.UFM_GetErrorString(ufs_res, strError);
if (ufs_res == p.UFS_OK) {
    // UFM_GetErrorString is succeeded
}
else {
    // UFM_GetErrorString is failed
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_GetTemplateType

Gets the parameter value.

Syntax

```
int UFM_GetTemplateType(long hMatcher,
                        int[] nTemplateType );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- pValue [out]: Receives parameter value of specified parameter type;
  pValue must point to adequate storage type matched to template type

Template type	Code	Description
UFM_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFM_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFM_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFM_STATUS

Remarks

Examples

```
int ufm_res;
long[] hMatcher = new long[1];
int[] nTemplateType = new int[1];

// make class library instance(BioMiniSDK p)

// Create hMatcher handle

ufm_res = p.UFM_GetTemplateType(hMatcher[], nTemplateType );
// Error handling routine is omitted
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_SetTemplateType

Gets parameter value.

Syntax

```
int UFM_SetTemplateType(long hMatcher,
                        int nTemplateType );
```

Parameters

- hMatcher [in]: Handle to the matcher object
- nTemplateType [in]: Parameter type; one of template type

Template type	Code	Description
UFM_TEMPLATE_TYPE_SUPREMA	2001	Suprema template type
UFM_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFM_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Return value

UFM_STATUS

Remarks

Examples

```
int ufm_res;
long[] hMatcher = new long[1];
int nTemplateType;

// make class library instance(BioMiniSDK p)

// Create hMatcher
handle nTemplateType =p.UFM_TEMPLATE_TYPE_SUPREMA;

ufm_res = p.UFM_SetTemplateType(hMatcher[], nTemplateType);
// Error handling routine is omitted
```

Requirements

Header	UFMatcher.dll
Library	UFMatcher.lib

See also

UFM_STATUS

UFM_STATUS

Every function in a UFMatcher module returns UFM_OK when it succeeds. When it fails, it returns a value corresponding to an error code. Please find the error code on the followings if you'd like to know the information about the UFM_STATUS (integer) value.

Code	Value	Description
UFM_OK	0	Success
UFM_ERROR	-1	General error
UFM_ERR_NO_LICENSE	-101	System has no license
UFM_ERR_LICENSE_NOT_MATC	-102	License does not match
UFM_ERR_LICENSE_EXPIRED	-103	License has expired
UFM_ERR_NOT_SUPPORTED	-111	This function is not supported
UFM_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFM_ERR_DATA_CORRUPTED	-113	Template data corrupted
UFM_ERR_NOT_INITIALIZED	-114	Module is not initailized
UFM_ERR_MATCH_TIMEOUT	-401	Matching is timeout
UFM_ERR_MATCH_ABORTED	-402	Matching is aborted
UFM_ERR_TEMPLATE_TYPE	-411	Template type does not match
UFM_ERR_TEMPLATE_FORMAT	-412	Template format does not match

Chapter 5. PREDEFINED VALUES

5.1. UFS_STATUS

Every function in the UFScanner module returns UFS_OK when it succeeds. When it fails, it returns a value corresponding to an error code. Please find the error code on the following table if you'd like to know the integer value of UFS_STATUS.

Code	Value	Description
UFS_OK	0	Success
UFS_ERROR	-1	General error
UFS_ERR_DEVICE_NOT_RESPOND	-10	Device is not responded
UFS_ERR_CAPTURE_TIMEOUT	-11	The time for capturing exceeds
UFS_ERR_USB_TIMEOUT	-12	The time for USB transfer exceeds
UFS_ERR_NO_LICENSE	-101	Device is not connected, or License is not located
UFS_ERR_LICENSE_NOT_MATCH	-102	License does not match
UFS_ERR_LICENSE_EXPIRED	-103	License has expired
UFS_ERR_NOT_SUPPORTED	-111	This function is not supported
UFS_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFS_ERR_UNKNOWN_TEMPLATE_FORMAT	-120	Abnormal template recognized
UFS_ERR_INVALID_ENCRYPTION	-121	Template buffer, encrypted template size is incorrect
UFS_ERR_INVALID_MODE	-122	Device is sleep mode
UFS_ERR_ALREADY_INITIALIZED	-201	Module is already initialized
UFS_ERR_NOT_INITIALIZED	-202	Module is not initialized
UFS_ERR_DEVICE_NUMBER_EXCEED	-203	Device number exceeds
UFS_ERR_LOAD_SCANNER_LIBRARY	-204	Error on loading the library of a scanner
UFS_ERR_CAPTURE_RUNNING	-211	Capturing is started using UFS_CaptureSingleImage or UFS_StartCapturing
UFS_ERR_CAPTURE_FAILED	-212	Capturing is timeout or aborted
UFS_ERR_FAKE_FINGER	-221	Fake finger is detected
UFS_ERR_FINGER_ON_SENSOR	-231	Remove finger from sensor
UFS_ERR_TIMEOUT	-241	Time out
UFS_ERR_NOT_GOOD_IMAGE	-301	Input image is not good
UFS_ERR_EXRACTION_FAILED	-302	Extraction is failed
UFS_ERR_ID_NOT_FOUND	-500	ID not found on device
UFS_ERR_NO_MATCH_FOUND	-501	Not registered fingerprint
UFS_ERR_NOT_IDENTICAL	-502	Not matched
UFS_ERR_CANNOT_ENROLL	-510	Cannot enroll fingerprint
UFS_ERR_NO_DATA	-511	No data found on device
UFS_ERR_NOT_AUTHORIZED	-520	Not authorized to use this API

If UFS_PARAM_DETECT_CORE is set as '1' (= use), please find the error code on the following table, too.

Code	Value	Description
UFS_ERR_CORE_NOT_DETECTED	-351	Core is not detected
UFS_ERR_CORE_TO_LEFT	-352	Move finger to left
UFS_ERR_CORE_TO_LEFT_TOP	-353	Move finger to left-top
UFS_ERR_CORE_TO_TOP	-354	Move finger to top
UFS_ERR_CORE_TO_RIGHT_TOP	-355	Move finger to right-top
UFS_ERR_CORE_TO_RIGHT	-356	Move finger to right
UFS_ERR_CORE_TO_RIGHT_BOTTOM	-357	Move finger to right-bottom
UFS_ERR_CORE_TO_BOTTOM	-358	Move finger to bottom
UFS_ERR_CORE_TO_LEFT_BOTTOM	-359	Move finger to left-bottom
UFS_ERR_FINGER_TOO_RIGHT	-401	Finger is too close to the right edge
UFS_ERR_FINGER_TOO_LEFT	-402	Finger is too close to the left edge
UFS_ERR_FINGER_TOO_TOP	-403	Finger is too close to the top
UFS_ERR_FINGER_TOO_BOTTOM	-404	Finger is too close to the bottom

5.2. UFM_STATUS

Every function in a UFMatcher module returns UFM_OK when it succeeds. When it fails, it returns a value corresponding to an error code. Please find the error code on the followings if you'd like to know the information about the UFM_STATUS (integer) value.

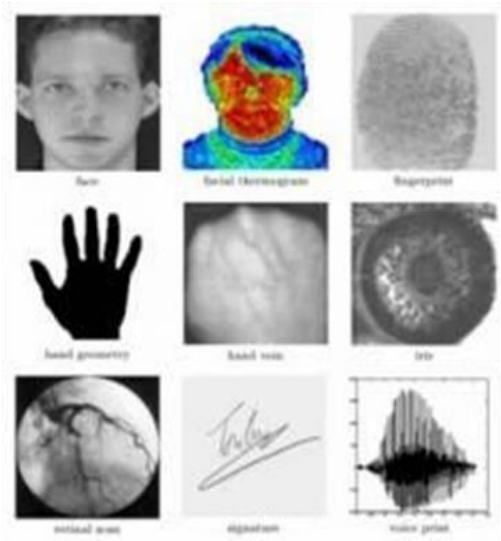
Code	Value	Description
UFM_OK	0	Success
UFM_ERROR	-1	General error
UFM_ERR_NO_LICENSE	-101	System has no license
UFM_ERR_LICENSE_NOT_MATCH	-102	License does not match
UFM_ERR_LICENSE_EXPIRED	-103	License has expired
UFM_ERR_NOT_SUPPORTED	-111	This function is not supported
UFM_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFM_ERR_DATA_CORRUPTED	-113	Template data corrupted
UFM_ERR_NOT_INITIALIZED	-114	Module is not initailized
UFM_ERR_MATCH_TIMEOUT	-401	Matching is timeout
UFM_ERR_MATCH_ABORTED	-402	Matching is aborted
UFM_ERR_TEMPLATE_TYPE	-411	Template type does not match
UFM_ERR_TEMPLATE_FORMAT	-412	Template format does not match

Chapter 6. APPENDIX
6.1. WHAT IS BIOMETRICS?

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

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

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



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

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

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

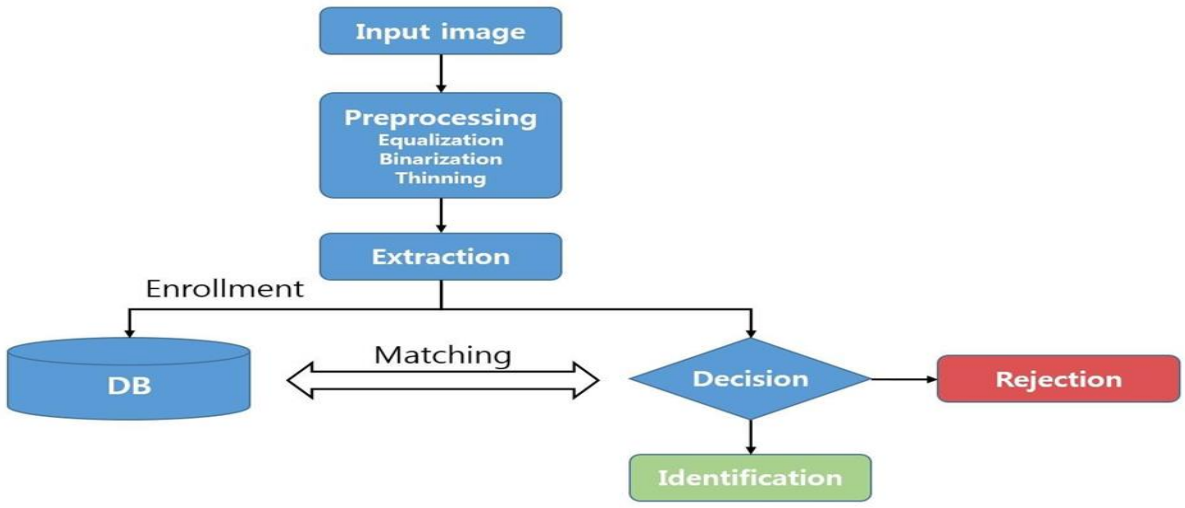
6.2. FINGERPRINT RECOGNITION

Classification

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

Basic format

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



Preprocessing

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

Feature Extraction

Fingerprint features can be performed as below methods; End point of ridge, minutiae placement, ridge direction pattern, connecting information between core and ridge, local feature containing ridge formation.
The feature extraction can be effected by fingerprint pressure, direction, location, placement, and condition of the fingerprint.

Feature Matching

The matching is comparing saved template pattern in the database with fingerprint features taken from the device. To determine the match, may use pattern matching, statistics identification, structure identification.

6.3. FAR, FRR and EER

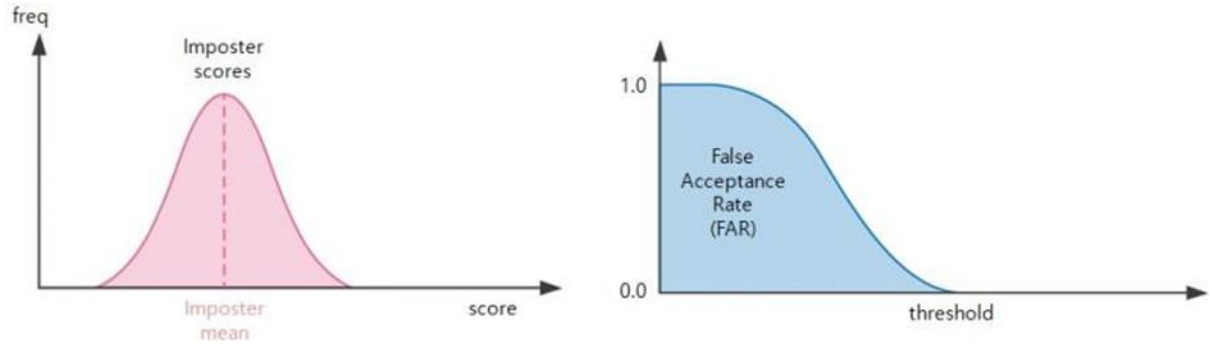
Here, we discuss some general principles of biometric recognition systems, describes different classification errors and explains how the quality of two systems can be compared objectively.

a. Identification vs. Verification

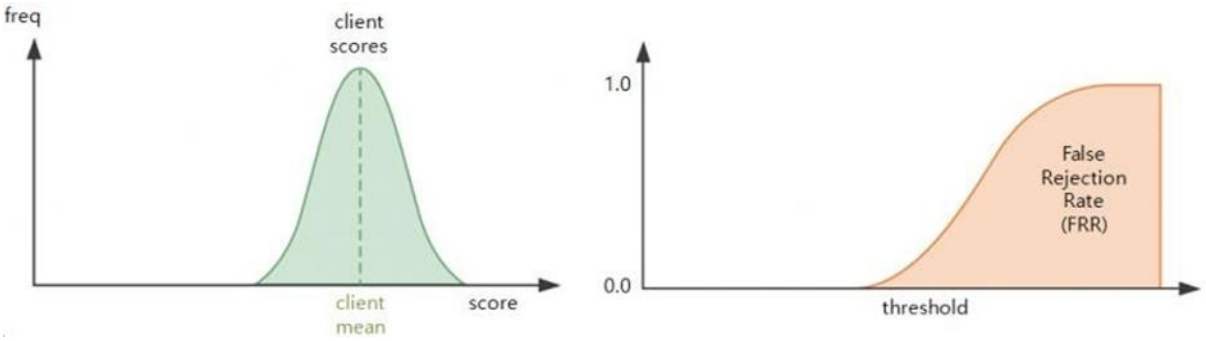
A biometric recognition system can run in two different modes: identification or verification. Identification is the process of trying to find out a person's identity by examining a biometric pattern calculated from the person's biometric features. In the identification case, the system is trained with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

b. Thresholding (False Acceptance / False Rejection)

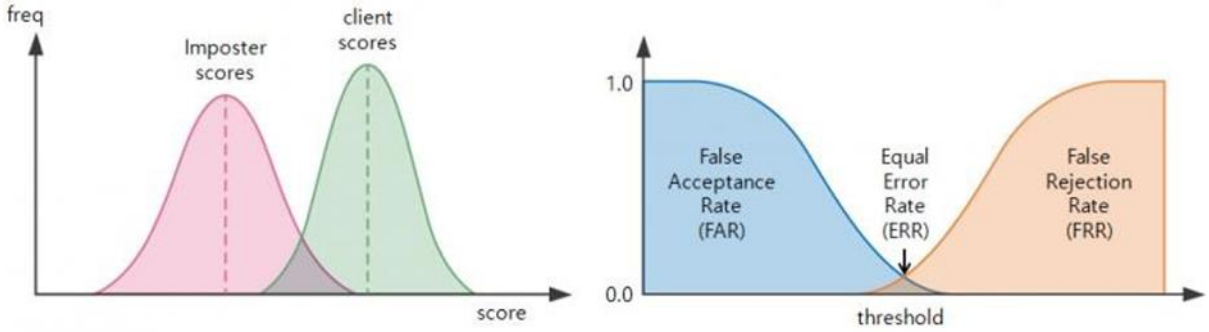
The threshold depending fraction of the falsely accepted patterns divided by the number of all impostor patterns is called **False Acceptance Rate (FAR)**. Its value is one, if all impostor patterns are falsely accepted and zero, if none of the impostor patterns is accepted. Look on the graphic on the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



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



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



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

6.4. SCANNERS

BioMini Slim 3

Ultra-slim FAP30 Authentication Scanner

BioMini Slim 3 is FAP30 certified fingerprint scanner featuring an array of cutting-edge technologies.
Along with its 15mm slim optical sensor, BioMini Slim 3 features Suprema's proprietary Multi-dynamic
Range(MDR) technology, FBI PIV/Mobile ID FAP30 compliance and Android device support. BioMini Slim 3
also provides developers more physical flexibility with its reduced form factor and the ultra-slim optical
sensor ensures robust operation over time. BioMini Slim 3 provides new and advanced Live Finger
Detection(LFD)technology by applying a machine learning method based AI(Artificial Intelligence) as an
World-class Trained Inspector which can be upgradeable by BioMini SDK.

Specification

	Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Sensing Area(20.32×25.4mm) ■ Image Size(400×500 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4) ■ IP Rating(IP65)
	Interface	■ USB(2.0 High-speed)
	Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification (CE, FCC, KC, CB, WHQL, IEC62471, WEEE) ■ Dimensions(83×49.5×20mm/WxLxH)
	Compatibility	■ Operating System (Windows, Linux, Android5.0 and Above)

BioMini Slim 2

Ultra-slim FAP20 Fingerprint scanner

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

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Sensing Area(15.24×20.32mm) ■ Image Size(300×400 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4) ■ IP Rating(IP65)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification (CE, FCC, KC, UL/CB, WHQL, IEC62471, WEEE) ■ Dimensions(72.8×40.7×18.5mm/WxLxH)
Compatibility	■ Operating System (Windows, Linux, Android5.0/Lolli Pop and Above)

BioMini Plus 2

FBI PIV and Mobile ID FAP20 Certified USB Fingerprint Scanner

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

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Sensing Area(16x18mm) ■ Image Size(315×354 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4) ■ IP Rating(IP65)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification(CE,FCC,KC,UL,WHQL, USB-IF,WEEE) ■ Dimensions(66×90x58mm/WxLxH)
Compatibility	■ Operating System (Windows, Linux, Android5.0/Lolli Pop and Above)

BioMini Slim

FBI PIV and Mobile ID FAP20 Certified USB Fingerprint Scanner

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

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Platen Size(18×25.4mm) ■ Sensing Area(17x25mm) ■ Image Size(320×480 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4) ■ IP Rating(IP65)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification(CE,FCC,KC,UL,WHQL, USB-IF,WEEE) ■ Dimensions(82×57.7x27mm/WxLxH)
Compatibility	■ Operating System (Windows, Linux, Android5.0/Lolli Pop and Above)

BioMini Combo

Contact Smart Card Reader with USB Fingerprint Scanner

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

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Platen Size(18×25.4mm) ■ Sensing Area(17x25mm) ■ Image Size(320×480 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4)
Card Support	■ Card Type(ISO 7816/EMV 2000with SAM Slot optical)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification(CE,FCC,KC,UL,WHQL, USB-IF,WEEE) ■ Dimensions(82×57.7x27mm/WxLxH)
Compatibility	■ Operating System (Windows, Linux, Android5.0/Lolli Pop and Above)

BioMini Plus

FBI PIV Approved Authentication Scanner

Suprema BioMiniPlus has been designed to provide high level security solution with its proven reliability through FBI-PIV certification. BioMini Plus features hybrid live finger detection (LFD) technology and multi award-winning Suprema Algorithm. Packed in a sleek and ergonomic design, it features durable 500dpi optical sensor and high speed USB 2.0 interface. Combined with its comprehensive SDK solution,

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Sensing Area(15.5×18.8mm) ■ Image Size(260×340 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4) ■ IP Rating(IP65)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification(CE,FCC,KC,UL,WHQL, USB-IF,WEEE) ■ Dimensions(66x90x58mm/WxLxH)
Compatibility	■ Operating System (Windows, Linux, Android5.0/Lolli Pop and Above)

BioMini

High Performance Authentication Scanner

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

Specification

Main	■ Sensor Type(Optical) ■ Resolution(500dpi/256gray) ■ Sensing Area(16x18mm)) ■ Image Size(288×320 pixels) ■ Compression Standards(WSQ) ■ Template Format(Suprema, ISO19794-2, ANSI-378) ■ Image Format(ISO19794-4)
Interface	■ USB(2.0 High-speed)
Hardware	■ Operating Temperature(-10°C ~ 50°C) ■ Certification(CE,FCC,KC,WHQL) ■ Dimensions(66x90x58mm/WxLxH)
Compatibility	■ Operating System (Windows, Linux, Android5.0/Lolli Pop and Above)

6.5. TROUBLESHOOTING

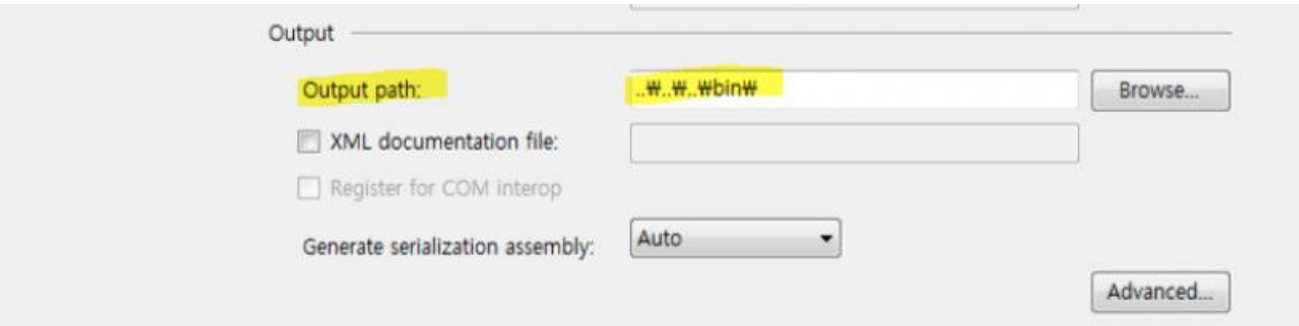
Q. When I tried the 'init' on the sample, i faced "ERROR: No license" error (No licensed device or file found).

A. In order to use the BioMini SDK, a scanner should be connected to PC. Please install the driver from <BioMini SDK_HOME>\install\drivers\SFR Driver(unified), and then connect the scanner.

Q. I have an error "Unable to load DLL 'UFScanner.dll': The specified module could not be found. (Exception from HRESULT: 0x8007007E)". How can I solve the error?

A. Please try to do the following way

1. Check a output path on Reference in Visual Studio as below image.



2. Confirm whether there is the 'UFScanner.dll' file on the output path. If there isn't, put the 'UFScanner.dll' file into the output path. 3. Try doing it again.
3. Try doing it again.

Q. I have tried copying all dlls from \bin folder to the bin directory of my project. But it gives the given error: "System.BadImageFormatException: An attempt was made to load a program with an incorrect format. (Exception from HRESULT: 0x8007000B)".

A. You can get the "0x8007000B" error since you used the dll files for 32bit in the 64bit environment, so please put following dll files from \bin\x64 folder into \bin directory of your project and re-build it.

- UFScanner.dll
- UFMatcher.dll
- Suprema.UFScanner.dll (This is a wrapper dll. Please copy this if you use the C# language.)
- Suprema.UFMatcher.dll (This is a wrapper dll. Please copy this if you use the C# language.)

Q. When I try to execute a diagnosis tool, i got an error "Unable to start program because mfc120u.dll or msvcr120.dll is not present in your computer".

A. Please download Visual C++ 2013 Redistributable package on the following link according to their environment. https://www.microsoft.com/en-US/download/details.aspx?id=40784

Q. When I click "Create log" on a diagnosis tool, this is crashed.

A. Please run the diagnosis tool as administrator.

Q. [Web agent] When I launched 'BioMini_WebAgent.exe' application, i got “Web-Agent server failed to start" error.

A. If ‘Biomini_WebAgent.exe’ application failed to launch and print “Web-Agent server failed to start” error on the screen, please check the following file’s existence where ‘Biomini_WebAgent.exe’ installed.

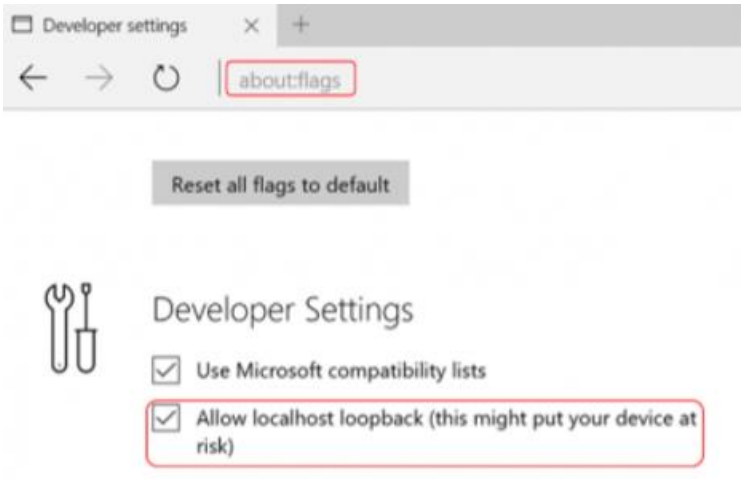
- ssleay32.dll, libeay32.dl

Q. [Web agent] I'm facing “network error 0x2efd" error in Microsoft Edge browser. (Windows 10)

A. Make sure ‘localhost loopback’ option is enabled. - Run ‘cmd.exe' with Administrator authority, and execute following script.

- CheckNetIsolation.exeLoopbackExempt -a -n=Microsoft.MicrosoftEdge_8wekyb3d8bbwe

- Type about:flags in the address edit box, and check if “Allow localhost loopback” option is enabled.



Q. [Web agent] I get “This Connection is Untrusted” error on Firefox browser.

A. Click “I Understand the Risks” and then click “Add Exception” button.