

Suprema PC SDK 3.4.1

Table Of Contents

Suprema PC SDK Reference Manual	1
Chapter 1. Introduction	2
Modules	2
Products	2
Licensing.....	2
Supported development tools and languages.....	3
Development system requirements	3
Chapter 2. What's new.....	4
Version 3.4.1	4
History.....	5
Version 3.4.....	5
Version 3.3.1	5
Version 3.3.0.....	5
Version 3.2.0	5
Version 3.1.0	5
Version 3.1.0.6	6
Version 3.0.0	6
Version 3.0.0.15	6
Version 3.0.0.14	6
Version 3.0.0.13	6
Version 3.0.0.12	6
Version 3.0.0.11	6
Version 3.0.0.8	6
Version 3.0.0.7	7
Version 3.0.0.5	7
Version 3.0.0.4	7
Chapter 3. Tutorial	8
Enroll fingerprints from scanner	9
0. Required product	9
1. Preliminaries.....	9
2. Initialize scanner module and check number of scanners.....	10
3. Get first scanner.....	12
4. Set parameters	13
5. Capture image and extract template.....	15
6. Uninitialize scanner module.....	17
Verification	19
0. Required product	19
1. Preliminaries.....	19
2. Create matcher.....	20
3. Set parameters	21
4. Verify	22
5. Delete matcher	25
Identification	27
0. Required product	27
1. Preliminaries.....	27
2. Create matcher.....	28
3. Set parameters	29

4. Identify	31
5. Delete matcher	35
Manage database	36
0. Required product	36
1. Preliminaries.....	36
2. Open database.....	37
3. Add one database entry.....	38
4. Get first database entry	40
5. Close database	41
Enroll fingerprints from image	43
0. Required product	43
1. Preliminaries.....	43
2. Create extractor.....	44
3. Set parameters	45
4. Load image and extract template.....	46
5. Delete matcher	48
Java Installation guide	49
0. Required product	49
1. Install Java SDK	49
2. Set Classpath	49
3. Build and run sample	49
Sample Demo Source guide.....	50
UFE30_Demo.....	50
UFE30_ImageDemo	50
UFE30_DatabaseDemo.....	50
UFE30_EnrollDemo	51
UFE30_MultiScannerDemo.....	51
Fake fingerprint detection.....	52
0. Required product	52
1. Set fake finger detection option.....	52
2. Use fake finger detection	52
Chapter 4. Sample	53
Before running sample applications	53
UFE30_Demo	54
Required products.....	54
Available languages.....	54
UFE30_Demo_Usage	55
Executable File Location	55
Required products.....	55
Picture of the Demo sample applications.....	55
User Interface Components.....	55
Scanner Initialization.....	58
Description	58
Using functions.....	58
Source Code.....	58
Update Scanner	63
Description	63
Using functions.....	63
Source Code.....	63

Scanner Uninitialization	65
Description	65
Using functions	65
Source Code.....	65
Capturing Option.....	68
Description	68
Using functions.....	68
Source Code.....	68
Start Capturing	71
Description	71
Using functions.....	71
Source Code.....	71
Abort Capturing	74
Description	74
Using functions.....	74
Source Code.....	74
Capture Single.....	76
Description	76
Using functions.....	76
Source Code.....	76
Template Extraction	78
Description	78
Using functions.....	78
Source Code.....	78
Enrollment	82
Description	82
Using functions.....	82
Source Code.....	82
Matching Option	94
Description	94
Using functions.....	94
Source Code.....	94
Verification	97
Description	97
Using functions.....	97
Source Code.....	97
Identification	105
Description	105
Using functions.....	105
Source Code.....	105
UFE30_ImageDemo	114
Required products.....	114
Available languages.....	114
UFE30_ImageDemo_Usage	115
Executable File Location	115
Required products.....	115
Picture of the ImageDemo sample applications	116
User Interface Component	119
Load Image File	122

Description	122
Using functions.....	122
Source Code.....	122
Scanner Setting.....	126
Description	126
Using functions.....	126
Source Code.....	126
Template Type Setting.....	128
Description	128
Using functions.....	128
Source Code.....	128
Extract a template from displayed image	130
Description	130
Using functions.....	130
Source Code.....	130
Extract a template from selecting image	135
Description	135
Using functions.....	135
Source Code.....	135
Match Two Images.....	142
Description	142
Using functions.....	142
Source Code.....	142
UFE30_DatabaseDemo	152
Required products.....	152
Available languages.....	152
UFE30_DatabaseDemo_Usage.....	153
Executable File Location	153
Required products.....	153
Picture of the Demo sample applications.....	153
User Interface Components.....	154
Initialization with Database Sample.....	157
Description	157
Using functions.....	157
Source Code.....	157
Uninitialization.....	164
Description	164
Using functions.....	164
Source Code.....	164
Enrollment with Database.....	168
Description	168
Using functions.....	168
Source Code.....	168
Identification with Database	172
Description	172
Using functions.....	172
Source Code.....	172
Verification with Database	177
Description	177

Using functions	177
Source Code.....	177
Delete All Database Data.....	181
Description	181
Using functions.....	181
Source Code.....	181
Delete Database Data	183
Description	183
Using functions.....	183
Source Code.....	183
Update User Information	185
Description	185
Using functions.....	185
Source Code.....	185
Update User Template	188
Description	188
Using functions.....	188
Source Code.....	188
UFE30_EnrollDemo.....	191
Required products.....	191
Available languages.....	191
UFE30_EnrollDemo_Usage	192
Executable File Location	192
Required products.....	192
Picture of the Demo sample applications.....	193
User Interface Components.....	195
UFE30_MultiScannerDemo	197
Required products.....	197
Available languages.....	197
UFE_MultiScannerDemo	198
UFE30_MultiScanner_Usage	198
Executable File Location	198
Required products.....	198
Picture of the Demo sample applications.....	199
User Interface Components.....	200
MultiScanner Initialization	202
Description	202
Using functions.....	202
Source Code.....	202
MultiScanner Uninitialization.....	204
Description	204
Using functions.....	204
Source Code.....	204
MultiScanner Enrollment.....	206
Description	206
Using functions.....	206
Source Code.....	206
MultiScanner Identification	209
Description	209

Using functions	209
Source Code.....	209
Chapter 5. Reference	212
UFScanner module	213
Requirements.....	213
Supported scanners	213
Definitions.....	214
Status return value (UFS_STATUS).....	214
Parameters	215
Template type	215
Scanner type	215
Scanner handle.....	216
Scanner callback function	216
Capture callback function.....	216
UFS_Init	218
UFS_Update	220
UFS_Uninit	222
UFS_SetScannerCallback	224
UFS_RemoveScannerCallback	227
UFS_GetScannerNumber.....	229
UFS_GetScannerHandle	231
UFS_GetScannerHandleByID.....	234
UFS_GetScannerIndex	237
UFS_GetScannerID	240
UFS_GetScannerType	243
UFS_GetParameter.....	246
UFS_SetParameter.....	250
UFS_IsSensorOn	254
UFS_IsFingerOn.....	257
UFS_CaptureSingleImage	260
UFS_StartCapturing	262
UFS_IsCapturing	265
UFS_AbortCapturing.....	268
UFS_Extract	271
UFS_SetEncryptionKey.....	275
UFS_EncryptTemplate	278
UFS_DecryptTemplate	282
UFS_GetCaptureImageBufferInfo	286
UFS_GetCaptureImageBuffer	289
UFS_DrawCaptureImageBuffer.....	293
UFS_DrawCaptureImageBuffer	296
UFS_SaveCaptureImageBufferToBMP	299
UFS_SaveCaptureImageBufferTo19794_4	302
UFS_SaveCaptureImageBufferToWSQ	305
UFS_ClearCaptureImageBuffer	308
UFS_GetErrorString	310
UFS_GetTemplateType	312
UFS_SetTemplateType	314
UFS_SelectTemplate	316

UFMatcher module.....	319
Requirements.....	319
Definitions.....	320
Status return value (UFM_STATUS)	320
Template type	320
Parameters	320
Matcher handle	321
UFM_Create.....	322
UFM_Delete.....	324
UFM_GetParameter.....	326
UFM_SetParameter	329
UFM_Verify.....	332
UFM_Identify, UFM_IdentifyMT.....	336
UFM_AbortIdentify.....	340
UFM_IdentifyInit	343
UFM_IdentifyNext	346
UFM_RotateTemplate	351
UFM_GetErrorString.....	354
UFM_GetTemplateType.....	356
UFM_SetTemplateType	358
UFExtractor module	360
Requirements.....	360
Definitions.....	361
Status return value (UFE_STATUS)	361
Template type	361
Parameters	361
Mode	362
Extractor handle	362
UFE_Create.....	363
UFE_Delete.....	365
UFE_GetMode	367
UFE_SetMode	369
UFE_GetParameter.....	371
UFE_SetParameter	374
UFE_DrawImage.....	377
UFE_Extract.....	379
UFE_SetEncryptionKey	382
UFE_EncryptTemplate	384
UFE_DecryptTemplate	387
UFE_LoadImageFromBMPFile.....	390
UFE_LoadImageFromBMPBuffer.....	393
UFE_LoadImageFromWSQFile.....	396
UFE_LoadImageFromWSQBuffer	398
UFE_GetErrorString.....	400
UFE_GetTemplateType	402
UFE_GetImageBufferTo19794_4ImageBuffer	404
UFE_GetImageBufferTo1BMPIImageBuffer.....	406
UFE_GetImageBufferToJPEGImageBuffer	408
UFE_GetImageBufferToJP2ImageBuffer	410

UFDatabase module	412
Requirements.....	412
Database table structure	412
Definitions	413
Status return value (UFD_STATUS).....	413
Database handle	413
UFD_Open	414
UFD_Close.....	416
UFD_AddData.....	418
UFD_UpdateDataByUserInfo	420
UFD_UpdateDataBySerial.....	423
UFD_RemoveDataByID.....	425
UFD_RemoveDataByUserInfo	427
UFD_RemoveDataBySerial	429
UFD_RemoveAllData	431
UFD_GetDataNumber	433
UFD_GetDataByIndex	435
UFD_GetDataByUserInfo	438
UFD_GetDataBySerial	440
UFD_GetTemplateNumber.....	443
UFD_GetTemplateListWithSerial.....	445
UFD_GetErrorString	448
Chapter 6. Reference (.NET)	449
Suprema.UFScanner module	450
Requirements.....	450
Supported scanners	450
Suprema.....	451
Suprema namespace.....	451
UFS_STATUS enumeration.....	452
UFS_SCANNER_TYPE enumeration.....	453
UFS_SCANNER_PROC delegate.....	454
UFS_CAPTURE_PROC delegate	455
UFScannerManagerScannerEventArgs class	456
UFScannerCaptureEventArgs class	457
UFScannerManager class	458
UFScannerManager constructor	459
Scanners property	460
ScannerEvent event	461
Init method	462
Update method	463
Uninit method	464
UFScannerManager.ScannerList class	465
Count property	466
Item property.....	467
UFScanner class	468
CaptureEvent event	470
ID property.....	471
Timeout property.....	472
Brightness property	473

Sensitivity property	474
DetectFake property	475
Serial property.....	476
DetectCore property	477
TemplateSize property.....	478
UseSIF property	479
ScannerType property	480
IsSensorOn property.....	481
IsFingerOn property	482
IsCapturing property.....	483
Handle property.....	484
SetScanner method.....	485
CaptureSingleImage method.....	486
StartCapturing method.....	487
AbortCapturing method.....	488
Extract method	489
SetEncryptionKey method.....	490
EncryptTemplate method.....	491
DecryptTemplate method	492
GetCaptureImageBuffer method.....	493
DrawCaptureImageBuffer method.....	494
DrawFeatureImageBuffer method	495
SaveCaptureImageBufferToBMP method.....	496
SaveCaptureImageBufferToTIF method.....	497
SaveCaptureImageBufferToJPG method	498
ClearCaptureImageBuffer method	499
GetErrorString method	500
SaveCaptureImageBufferTo19794_4 method	501
SelectTemplate method	502
Suprema.UFMatcher module	503
Requirements.....	503
Suprema.....	504
Suprema namespace.....	504
UFM_STATUS enumeration	505
UFMatcher class.....	506
FastMode property	507
SecurityLevel property	508
UseSIF property	509
Verify method	510
Identify, IdentifyMT method	511
AbortIdentify method	513
IdentifyInit method.....	514
IdentifyNext method.....	515
RotateTemplate method.....	516
GetErrorString method	517
Suprema.UFExtractor module	518
Requirements.....	518
Suprema.....	519
Suprema namespace.....	519

UFE_STATUS enumeration	520
UFE_MODE enumeration	521
UFExtractor class	522
Mode property	523
DetectCore property	524
TemplateSize property	525
UseSIF property	526
Extract method	527
SetEncryptionKey method	528
EncryptTemplate method	529
DecryptTemplate method	530
LoadImageFromBMPFile method	531
LoadImageFromTIFFFile method	532
LoadImageFromJPGFile method	533
LoadImageFromBMPBuffer method	534
LoadImageFromWSQFile method	535
LoadImageFromWSQBuffer method	536
GetErrorString method	537
GetImageBufferTo19794_4ImageBuffer	538
GetImageBufferToBMPImageBuffer	540
GetImageBufferToJPEGImageBuffer	542
GetImageBufferToJP2ImageBuffer	544
Suprema.UFDatabase module	546
Requirements	546
Database table structure	546
Suprema	547
Suprema namespace	547
UFD_STATUS enumeration	548
UFDatabase class	549
Open method	550
Close method	551
AddData method	552
UpdateDataByUserInfo method	553
UpdateDataBySerial method	554
RemoveDataByUserID method	555
RemoveDataByUserInfo method	556
RemoveDataBySerial method	557
RemoveAllData method	558
GetDataNumber method	559
GetDataByIndex method	560
GetDataByUserInfo method	561
GetDataBySerial method	562
GetTemplateListWithSerial method	563
GetErrorString method	564
FAQ	565
System Issue	565
Function & Parameter Issue	565
Template Issue	566
Scanner Issue	566

Suprema PC SDK 3.4.1

DataBase Issue	567
Performance Issue	567
Appendix A. Contacts	569
Headquarters	569
E-mail support.....	569
If you wish to directly query a problem in the Suprema PC SDK, send a mail to support@suprema.com or sales@suprema.com. Appendix B. Distribution Content	569
Appendix B. Distribution Content	570
bin.....	571
docs.....	573
include	574
install	575
lib.....	576
samples	577
Index.....	578

Suprema PC SDK Reference Manual

Version 3.4.1

Copyright (C) 2012 Suprema Inc.

Table of Contents

- [1. Introduction](#)
- [2. What's new](#)
- [3. History](#)
- [4. Tutorial](#)
- [5. Sample](#)
- [6. Reference](#)
- [7. Reference \(.NET\)](#)
- [A. Contacts](#)
- [B. Distribution Content](#)
- [C. FAQ](#)



Chapter 1. Introduction

Modules

Suprema PC SDK consists of UFScanner, UFMatcher, UFExtractor, and UFDatabase modules and sample applications to describe how to use these modules. The main usage of each module is summarized as follows,

Module name	Main usage
UFScanner	managing scanners, capturing finger images from scanners, extracting templates from captured images using scanners
UFMatcher	verifying fingerprints using two templates, identifying fingerprints using the template array
UFExtractor	extracting templates from input images
UFDatabase	managing database, adding / updating / removing / getting templates with user data

Products

The qualification of using each module depends on products and Suprema PC SDK is divided into following products,

Product name	Supported modules
Suprema BioMini Enroll SDK	UFScanner
Suprema BioMini SDK	UFScanner, UFMatcher, UFDatabase(not supported on java)
Suprema Match SDK	UFMatcher
Suprema Image SDK	UFMatcher, UFExtractor

Licensing

Every Suprema PC SDK module checks a license file when it is initialized. The license file is named as **UFLicense.dat** and this file **should be located in the same directory with Suprema PC SDK modules**. But Universal manufacturer ID embedded in scanners without license control. By the licensing policy, the license file may require the key lock provided by Suprema Inc. For the detailed licensing policy, contact Suprema Inc.

Supported development tools and languages

- Visual Studio 6.0 - Visual C++, Visual Basic 6.0
- Visual Studio 2003 (7.1) (.NET Framework 1.1) - Visual C++, Visual C#, Visual Basic .NET
- Visual Studio 2005 (8.0) (.NET Framework 2.0) - Visual C++, Visual C#, Visual Basic .NET
- Java SDK 1.4 or higher (using JNA(Java Native Access))

Development system requirements

- Pentium-compatible 500MHz processor or better
- Microsoft Windows 2000 / 2003 / 2008 / XP / Vista / 7
- Fingerprint scanner driver for using UFScanner module

For information about how to use Suprema PC SDK module see Tutorial. Please check basic usage in tutorial chapter.

If you have any question about SDK, please see [FAQ](#) page first. Please check questions and answers related to your issue.

Chapter 2. What's new

Version 3.4.1

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

History

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 fake finger detection algorithm.
- Support WSQ file format in Image SDK - UFExtractor
- Support image drawing function in Image SDK - UFExtractor
- 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

- Add 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

- Add license scheme Using RSD(Real Scan Device)
- UFExtractor - Modify maximum image size 1024 x 1024 from 640 x 480

Version 3.0.0.14

- UFScanner, UFExtractor - Add function which get number of Minutiae
 - UFS_GetFeatureNumber, UFE_GetFeatureNumber
- UFMatcher - Add 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

- UFExtractor - Add UFE_MODE_GENERAL mode (Using general image type)
- 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

Chapter 3. Tutorial

This chapter describes the basic usages of Suprema PC SDK functions in a step-by-step manner. The usages include following topics,

Topics
<u>Enroll fingerprints from scanner</u>
<u>Verification</u>
<u>Identification</u>
<u>Manage database</u>
<u>Enroll fingerprints from image</u>
<u>Fake finger detection</u>
<u>Sample source guide</u>

Enroll fingerprints from scanner

0. Required product

[Suprema BioMini Enroll SDK](#) or [Suprema BioMini SDK](#)

1. Preliminaries

Visual C++

```
// Add Suprema UFScanner lib (lib\UFScanner.lib) to the
Project
// Add following statements in the source
#include "UFScanner.h"

// We use 384 bytes template size in this tutorial
#define TEMPLATE_SIZE 384
```

Visual Basic 6.0

```
' Add Suprema type library (bin\Suprema.tlb) using browse
button in the References dialog (drop down the Project menu
and select the References item)

Option Explicit
Option Base 0

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Long = 384
```

Visual C#

```
// Add Suprema UFScanner library (bin\Suprema.UFScanner.dll)
using browse tap in the Add References dialog
// Add following statements in the source
using Suprema

// We use 384 bytes template size in this tutorial
const int MAX_TEMPLATE_SIZE = 384;
```

Visual Basic .NET

```
' Add Suprema UFScanner library (bin\Suprema.UFScanner.dll)
using browse tap in the Add References dialog
' Add following statements in the source
Imports Suprema

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Integer = 384
```

```
java
//import Suprema java package
import com.suprema.ufs31.*;

//Add following statements in the source
//We use 384 bytes template size in this tutorial
public final int MAX_TEMPLATE_SIZE=384;
```

2. Initialize scanner module and check number of scanners

Visual C++
<pre>UFS_STATUS ufs_res; int nScannerNumber; // Initialize 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 the tutorial, we omit error check codes // Check number of scanners ufs_res = UFS_GetScannerNumber(&nScannerNumber); // If number of scanner is under one, that means there is no scanner in this system</pre>

Visual Basic 6.0
<pre>Dim ufs_res As UFS_STATUS Dim nScannerNumber As Integer ' Initialize 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 the tutorial, we omit error check codes ' Check number of scanners ufs_res = UFS_GetScannerNumber(ScannerNumber) ' If number of scanner is under one, that means there is no scanner in this system</pre>

Visual C#
<pre>UFS_STATUS ufs_res; UFScannerManager ScannerManager; int nScannerNumber;</pre>

```
// Create an instance of ScannerManager
ScannerManager = new UFScannerManager(this);

// Initialize scanner module
ufs_res = ScannerManager.Init();
// Always check status return codes after running SDK
functions
// Meaning of status return code can be retrieved using
UFScanner.GetErrorString()
// In the tutorial, we omit error check codes

// Check number of scanners
nScannerNumber = ScannerManager.Scanners.Count;
```

Visual Basic .NET

```
Dim ufs_res As UFS_STATUS
Dim ScannerManager As UFScannerManager
Dim nScannerNumber As Integer

' Create an instance of ScannerManager
ScannerManager = New UFScannerManager(Me)

' Initialize scanner module
ufs_res = ScannerManager.Init()
' Always check status return codes after running SDK
functions
' Meaning of status return code can be retrieved using
UFScanner.GetErrorString()
' In the tutorial, we omit error check codes

' Check number of scanners
nScannerNumber = ScannerManager.Scanners.Count
```

java

```
public UFScannerClass libScanner=null;

//load UFScanner.dll (UFScannerClass.class is wrapper class
//that mapping to UFscanner.dll)
libScanner =
(UFScannerClass)Native.loadLibrary("UFScanner",UFScannerClas
s.class);

//Initialize scanner module
int nRes =0;
nRes = libScanner.UFS_Init();

//Always check status return codes after running SDK
functions
```

```
//Meaning of status return code can be retrieved using
UFScanner.GetErrorString()
//In the tutorial, we omit error check codes
byte[] refErr = new byte[512];
nRes = libScanner.UFS_GetErrorString(nRes,refErr);
if(nRes==0){
    System.out.println("UFS_GetErrorString err is
"+Native.toString(refErr));
}

//Check number of scanners int nNumber=0;
int nRes=0;
int nNumber=0;
IntByReference refNumber = new IntByReference();
nRes = libScanner.UFS_GetScannerNumber(refNumber);
nNumber = refNumber.getValue();
```

3. Get first scanner

Visual C++

```
UFS_STATUS ufs_res;
HUFScanner hScanner;

// Get first scanner handle (0 means first scanner)
ufs_res = UFS_GetScannerHandle(0, &hScanner);
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long

' Get first scanner handle (0 means first scanner)
ufs_res = UFS_GetScannerHandle(nCurScannerIndex, hScanner)
```

Visual C#

```
// ScannerManager comes from section 2
UFScanner Scanner = null;

// Using first scanner (0 means first scanner)
Scanner = ScannerManager.Scanners[0];
```

Visual Basic .NET

```
' ScannerManager comes from section 2
Dim Scanner As UFScanner = Nothing

' Using first scanner (0 means first scanner)
Scanner = ScannerManager.Scanners(0)
```

java

```
//ScannerManager comes from section 2
Pointer hScanner =null;

//Using first scanner (0 means first scanner)
int nRes=0;
PointerByReference refScanner = new PointerByReference();
nRes = libScanner.UFS_GetScannerHandle(0,refScanner);

hScanner = refScanner.getValue();
```

4. Set parameters

Visual C++

```
// hScanner comes from section 3
UFS_STATUS ufs_res;
int value;

// Set timeout for capturing images to 5 seconds
value = 5000;
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT,
&value);

// Set template size to 384 bytes
value = MAX_TEMPLATE_SIZE;
ufs_res = UFS_SetParameter(hScanner,
UFS_PARAM_TEMPLATE_SIZE, &value);

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

Visual Basic 6.0

```
' hScanner comes from section 3
Dim ufs_res As UFS_STATUS
Dim value As Long

' Set timeout for capturing images to 5 seconds
value = 5000;
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT,
&value)

' Set template size to 384 bytes
value = MAX_TEMPLATE_SIZE;
ufs_res = UFS_SetParameter(hScanner,
UFS_PARAM_TEMPLATE_SIZE, &value)

' Set not to detect core when extracting template
value = FALSE;
```

Suprema PC SDK 3.4.1

```
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_DETECT_CORE,  
&value)
```

Visual C#

```
// Scanner comes from section 3  
  
// Set timeout for capturing images to 5 seconds  
Scanner.Timeout = 5000;  
  
// Set template size to 384 bytes  
Scanner.TemplateSize = MAX_TEMPLATE_SIZE;  
  
// Set not to detect core when extracting template  
Scanner.DetectCore = false;
```

Visual Basic .NET

```
' Scanner comes from section 3  
  
' Set timeout for capturing images to 5 seconds  
Scanner.Timeout = 5000;  
  
' Set template size to 384 bytes  
Scanner.TemplateSize = MAX_TEMPLATE_SIZE;  
  
' Set not to detect core when extracting template  
Scanner.DetectCore = false;
```

java

```
// hScanner comes from section 3  
int nRes;  
IntByReference pValue = new IntByReference();  
  
// Set timeout for capturing images to 5 seconds  
pValue.setValue(5000);  
int nRes =  
libScanner.UFS_SetParameter(hScanner, libScanner.UFS_PARAM_TIMEOUT,pValue);  
  
// Set brightness 100  
pValue.setValue(100);  
ufs_res=libScanner.UFS_SetParameter(hScanner,  
UFS_PARAM_BRIGHTNESS, pValue);  
  
// Set scanner sensitivity  
pValue.setValue(4);  
nRes =  
libScanner.UFS_SetParameter(hScanner, libScanner.UFS_PARAM_SENSITIVITY,pValue);
```

5. Capture image and extract template

Visual C++

```
// hScanner comes from section 3
UFS_STATUS ufs_res;
unsigned char Template[MAX_TEMPLATE_SIZE];
int TemplateSize;
int nEnrollQuality;

// Clear capture buffer
ufs_res = UFS_ClearCaptureImageBuffer(hScanner);

// Capture single image
ufs_res = UFS_CaptureSingleImage(hScanner);
// If capturing images is fail, iterate above capture
routine or show error message

// Extract template from captured image
ufs_res = UFS_Extract(hScanner, Template, &TemplateSize,
&nEnrollQuality);
// If extraction is succeed, check nEnrollQuality is above
predefined quality threshold
```

Visual Basic 6.0

```
' hScanner comes from section 3
Dim ufs_res As UFS_STATUS
Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateSize As Long
Dim EnrollQuality As Long

' Clear capture buffer
ufs_res = UFS_ClearCaptureImageBuffer(hScanner)

' Capture single image
ufs_res = UFS_CaptureSingleImage(hScanner)
' If capturing images is fail, iterate above capture routine
or show error message

' Extract template from captured image
ufs_res = UFS_Extract(hScanner, Template(0), TemplateSize,
EnrollQuality)
' If extraction is succeed, check nEnrollQuality is above
predefined quality threshold
```

Visual C#

```
// Scanner comes from section 3
UFS_STATUS ufs_res;
byte[] Template = new byte[MAX_TEMPLATE_SIZE];
int TemplateSize;
```

Suprema PC SDK 3.4.1

```
int EnrollQuality;

// Clear capture buffer
ufs_res = Scanner.ClearCaptureImageBuffer();

// Capture single image
ufs_res = Scanner.CaptureSingleImage();
// If capturing images is fail, iterate above capture
routine or show error message

// Extract template from captured image
ufs_res = Scanner.Extract(Template, out TemplateSize, out
EnrollQuality);
// If extraction is succeed, check nEnrollQuality is above
predefined quality threshold
```

Visual Basic .NET

```
' Scanner comes from section 3
Dim ufs_res As UFS_STATUS
Dim Template As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim TemplateSize As Integer = Nothing
Dim EnrollQuality As Integer = Nothing

' Clear capture buffer
ufs_res = Scanner.ClearCaptureImageBuffer()

' Capture single image
ufs_res = Scanner.CaptureSingleImage()
' If capturing images is fail, iterate above capture routine
or show error message

' Extract template from captured image
ufs_res = Scanner.Extract(Template, TemplateSize,
EnrollQuality)
' If extraction is succeed, check nEnrollQuality is above
predefined quality threshold
```

java

```
// hScanner comes from section 3
UFS_STATUS ufs_res;
unsigned char Template[MAX_TEMPLATE_SIZE];
int TemplateSize;
int nEnrollQuality;
int nRes;

// Clear capture buffer
libScanner.UFS_ClearCaptureImageBuffer(hScanner);

// Capture single image
```

```

nRes = libScanner.UFS_CaptureSingleImage(hScanner);
// If capturing images is fail, iterate above capture routine
or show error message

// Extract template from captured image
byte[] bTemplate = new byte[512];
IntByReference refTemplateSize = new IntByReference();
IntByReference refTemplateQuality = new IntByReference();

nRes =
libScanner.UFS_Extract(hScanner,bTemplate,refTemplateSize,refT
emplateQuality);
// If extraction is succeed, check nEnrollQuality is above
predefined quality threshold

```

6. Uninitialize scanner module

Visual C++

```

UFS_STATUS ufs_res;

// Uninitialize scanner module
ufs_res = UFS_Uninit();

```

Visual Basic 6.0

```

Dim ufs_res As UFS_STATUS

' Uninitialize scanner module
ufs_res = UFS_Uninit()

```

Visual C#

```

// ScannerManager comes from section 2
UFS_STATUS ufs_res;

// Uninitialize scanner module
ufs_res = ScannerManager.Uninit();

```

Visual Basic .NET

```

' ScannerManager comes from section 2
Dim ufs_res As UFS_STATUS

' Uninitialize scanner module
ufs_res = ScannerManager.Uninit()

```

java

```

int nRes;

// Uninitialize scanner module
nRes = libScanner.UFS_Uninit();

```


Verification

0. Required product

[Suprema BioMini SDK](#) or [Suprema Match SDK](#) or [Suprema Image SDK](#)

1. Preliminaries

Visual C++

```
// Add Suprema UFMatcher lib (lib\UFMatcher.lib) to the
Project
// Add following statements in the source
#include "UFMatcher.h"

// We use 384 bytes template size in this tutorial
#define MAX_TEMPLATE_SIZE 384
```

Visual Basic 6.0

```
' Add Suprema type library (bin\Suprema.tlb) using browse
button in the References dialog (drop down the Project menu
and select the References item)

Option Explicit
Option Base 0

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Long = 384
```

Visual C#

```
// Add Suprema UFMatcher library (bin\Suprema.UFMatcher.dll)
using browse tap in the Add References dialog
// Add following statements in the source
using Suprema

// We use 384 bytes template size in this tutorial
const int MAX_TEMPLATE_SIZE = 384;
```

Visual Basic .NET

```
' Add Suprema UFMatcher library (bin\Suprema.UFMatcher.dll)
using browse tap in the Add References dialog
' Add following statements in the source
Imports Suprema

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Integer = 384
```

```
java
//import Suprema java package
import com.suprema.ufe31.*;

// We use 384 bytes template size in this tutorial
public final int MAX_TEMPLATE_SIZE=384;
```

2. Create matcher

Visual C++

```
UFM_STATUS ufm_res;
HUFMatcher hMatcher;

// Create matcher
ufm_res = UFM_Create(&hMatcher);
// Always check status return codes after running SDK
functions
// Meaning of status return code can be retrieved using
UFM_GetErrorString()
// In the tutorial, we omit error check codes
```

Visual Basic 6.0

```
Dim ufm_res As UFM_STATUS
Dim hMatcher As Long

' Create matcher
ufm_res = UFM_Create(hMatcher)
' Always check status return codes after running SDK
functions
' Meaning of status return code can be retrieved using
UFM_GetErrorString()
' In the tutorial, we omit error check codes
```

Visual C#

```
UFMatcher Matcher;

// Create matcher
Matcher = new UFMatcher();
// Always check status return codes after running SDK
functions
// Meaning of status return code can be retrieved using
UFMatcher.GetErrorString()
// In the tutorial, we omit error check codes
```

Visual Basic .NET

```
Dim Matcher As UFMatcher
```

```
' Create matcher
Matcher = New UFMatcher()
' Always check status return codes after running SDK
functions
' Meaning of status return code can be retrieved using
UFMatcher.GetErrorString()
' In the tutorial, we omit error check codes
```

```
java
int nRes;
Pointer hMatcher = null;
PointerByReference refMatcher = new PointerByReference();
UFMatcherClass libMatcher=null;

//load UFMatcher.dll (UFMatcherClass.class 는 UFMatcher.dll에
//埋핑되는 wrapper class)
libMatcher =
(UFMatcherClass)Native.loadLibrary("UFMatcher",UFMatcherClass.
class);

// Create matcher
nRes = libMatcher.UMF_Create(refMatcher);
hMatcher = refMatcher.getValue();
// Always check status return codes after running SDK
functions
// Meaning of status return code can be retrieved using
libMatcher.UMF_GetErrorString()
// In the tutorial, we omit error check codes
```

3. Set parameters

```
Visual C++
// hMatcher comes from section 2
UFM_STATUS ufm_res;
int value;

// Set security level to 3
value = 3;
ufm_res = UFM_SetParameter(hMatcher,
UFM_PARAM_SECURITY_LEVEL, &value);
```

```
Visual Basic 6.0
' hMatcher comes from section 2
Dim ufm_res As UFM_STATUS

' Set security level to 3
ufm_res = UFM_SetParameter(hMatcher,
UFM_PARAM_SECURITY_LEVEL, 3)
```

Visual C#

```
// Matcher comes from section 2

// Set security level to 3
Matcher.SecurityLevel = 3;
```

Visual Basic .NET

```
' Matcher comes from section 2

' Set security level to 3
Matcher.SecurityLevel = 3
```

java

```
// hMatcher comes from section 2
int nRes;
IntByReference value = new IntByReference;

// Set security level to 3
pValue.setValue(3);
nRes =
libMatcher.UMF_SetParameter(hMatcher, libMatcher.UMF_PARAM_SECURITY_LEVEL, pValue);

//Set fast mode on
pValue.setValue(1);
nRes = libMatcher.UMF_SetParameter(hMatcher,
libMatcher.UMF_PARAM_FAST_MODE, pValue)
```

4. Verify

Visual C++

```
// hMatcher comes from section 2
UFM_STATUS ufm_res;
unsigned char Template1[MAX_TEMPLATE_SIZE];
int Template1Size;
unsigned char Template2[MAX_TEMPLATE_SIZE];
int Template2Size;
int bVerifySucceed;

// Get Template1 from scanner or image or database
// Get Template2 from scanner or image or database

// Verify two templates
ufm_res = UFM_Verify(hMatcher, Template1, Template1Size,
Template2, Template2size, &bVerifySucceed);

if (ufm_res != UFM_OK) {
```

```

        // Execute error handling codes
    } else {
        if (bVerifySucceed) {
            // Verification succeed
        } else {
            // Verification failed
        }
    }
}

```

Visual Basic 6.0

```

' hMatcher comes from section 2
Dim ufm_res As UFM_STATUS
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2size As Long
Dim VerifySucceed As Long

' Get Template1 from scanner or image or database
' Get Template2 from scanner or image or database

' Verify two templates
ufm_res = UFM_Verify(hMatcher, Template1(0), Template1Size,
Template2(0), Template2size, VerifySucceed);

If (ufm_res <> UFM_STATUS.OK) Then
    ' Execute error handling codes
Else
    If (VerifySucceed = 1) Then
        ' Verification succeed
    Else
        ' Verification failed
    End If
End If

```

Visual C#

```

// Matcher comes from section 2
UFM_STATUS ufm_res;
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
int Template1Size;
byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int Template2Size;
bool VerifySucceed;

// Get Template1 from scanner or image or database
// Get Template2 from scanner or image or database

// Verify two templates
ufm_res = Matcher.Verify(Template1, Template1Size,

```

Suprema PC SDK 3.4.1

```
Template2, Template2Size, out VerifySucceed);

if (ufm_res != UFM_STATUS.OK) {
    // Execute error handling codes
} else {
    if (VerifySucceed) {
        // Verification succeed
    } else {
        // Verification failed
    }
}
```

Visual Basic .NET

```
' Matcher comes from section 2
Dim ufm_res As UFM_STATUS
Dim Template1 As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim Template1Size As Integer = Nothing
Dim Template2 As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim Template2Size As Integer = Nothing
Dim VerifySucceed As Boolean = Nothing

' Get Template1 from scanner or image or database
' Get Template2 from scanner or image or database

' Verify two templates
ufm_res = Matcher.Verify(Template1, Template1Size,
Template2, Template2Size, VerifySucceed)

If (ufm_res <> UFM_STATUS.OK) Then
    ' Execute error handling codes
Else
    If (VerifySucceed = 1) Then
        ' Verification succeed
    Else
        ' Verification failed
    End If
End If
```

java

```
// hMatcher comes from section 2
int nRes;
byte Template1[MAX_TEMPLATE_SIZE];
IntByReference Template1Size = new IntByReference();
IntByReference refTemplateQuality= new IntByReference();
byte Template2[MAX_TEMPLATE_SIZE];
IntByReference Template2Size = new IntByReference();
IntByReference bVerifySucceed = new IntByReference;

// Get Template1 from scanner or image or database
```

```
// Get Template2 from scanner or image or database

nRes =
libScanner.UFS_Extract(hScanner,bTemplate,Template1Size,refTem
plateQuality);

nRes = libMatcher.UMF_Verify(hMatcher, Template1,
Template1Size.getValue(),
Template2, Template2Size.getValue(), bVerifySucceed );

if (nRes != UFM_OK) {
    // Execute error handling codes
} else {
    if (bVerifySucceed.getValue()) {
        // Verification succeed
    } else {
        // Verification failed
    }
}
```

5. Delete matcher

Visual C++

```
// hMatcher comes from section 2
UFM_STATUS ufm_res;

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

Visual Basic 6.0

```
' hMatcher comes from section 2
Dim ufm_res As UFM_STATUS

' Delete matcher
ufm_res = UFM_Delete(hMatcher)
```

Visual C#

```
// No explicit delete code is needed
```

Visual Basic .NET

```
' No explicit delete code is needed
```

java

```
//hMatcher comes from section 2
int nRes;

//Delete matcher
nRes= libMatcher.UMF_Delete(hMatcher);
```


Identification

0. Required product

[Suprema BioMini SDK](#) or [Suprema Match SDK](#) or [Suprema Image SDK](#)

1. Preliminaries

Visual C++

```
// Add Suprema UFMatcher lib (lib\UFMatcher.lib) to the
Project
// Add following statements in the source
#include "UFMatcher.h"

// We use 384 bytes template size in this tutorial
#define MAX_TEMPLATE_SIZE 384
// Set maximum template number to 50 (number depends on
application)
#define MAX_TEMPLATE_NUM 50
```

Visual Basic 6.0

```
' Add Suprema type library (bin\Suprema.tlb) using browse
button in the References dialog (drop down the Project menu
and select the References item)

Option Explicit
Option Base 0

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Long = 384
' Set maximum template number to 50 (number depends on
application)
Const MAX_TEMPLATE_NUM As Long = 50
```

Visual C#

```
// Add Suprema UFMatcher library (bin\Suprema.UFMatcher.dll)
using browse tap in the Add References dialog
// Add following statements in the source
using Suprema

// We use 384 bytes template size in this tutorial
const int MAX_TEMPLATE_SIZE = 384;
// Set maximum template number to 50 (number depends on
application)
const int MAX_TEMPLATE_NUM = 50;
```

Suprema PC SDK 3.4.1

Visual Basic .NET

```
' Add Suprema UFM matcher library (bin\Suprema.UFM matcher.dll)
using browse tap in the Add References dialog
' Add following statements in the source
Imports Suprema

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Integer = 384
' Set maximum template number to 50 (number depends on
application)
Const MAX_TEMPLATE_NUM = 50
```

java

```
//import Suprema java package
import com.suprema.ufe31.*;

//We use 384 bytes template size in this tutorial
//Set maximum template number to 50 (number depends on
application)

public final int MAX_TEMPLATE_SIZE=384;
public final int MAX_TEMPLATE_NUM=50;
```

2. Create matcher

Visual C++

```
UFM_STATUS ufm_res;
HUFMatcher hMatcher;

// Create matcher
ufm_res = UFM_Create(&hMatcher);
// Always check status return codes after running SDK
functions
// Meaning of status return code can be retrieved using
UFM_GetErrorString()
// In the tutorial, we omit error check codes
```

Visual Basic 6.0

```
Dim ufm_res As UFM_STATUS
Dim hMatcher As Long

' Create matcher
ufm_res = UFM_Create(hMatcher)
' Always check status return codes after running SDK
functions
' Meaning of status return code can be retrieved using
UFM_GetErrorString()
' In the tutorial, we omit error check codes
```

Visual C#

```
UFMatcher Matcher;

// Create matcher
Matcher = new UFMatcher();
// Always check status return codes after running SDK
functions
// Meaning of status return code can be retrieved using
UFMatcher.GetErrorString()
// In the tutorial, we omit error check codes
```

Visual Basic .NET

```
Dim Matcher As UFMatcher

' Create matcher
Matcher = New UFMatcher()
' Always check status return codes after running SDK
functions
' Meaning of status return code can be retrieved using
UFMatcher.GetErrorString()
' In the tutorial, we omit error check codes
```

java

```
int nRes;
Pointer hMatcher = null;
PointerByReference refMatcher = new PointerByReference;
UFMatcherClass libMatcher=null;

//load library(libMatcher)
libMatcher =
(UFMatcherClass)Native.loadLibrary("UFMatcher",UFMatcherClass.
class);

// Create matcher
nRes = libMatcher.UMF_Create(refMatcher);
hMatcher = refMatcher.getValue();
// Always check status return codes after running SDK
functions
// Meaning of status return code can be retrieved using
libMatcher.UMF_GetErrorString()
// In the tutorial, we omit error check codes
```

3. Set parameters

Visual C++

```
// hMatcher comes from section 2
UFM_STATUS ufm_res;
int value;
```

Suprema PC SDK 3.4.1

```
// Set security level to 4
value = 4;
ufm_res = UFM_SetParameter(hMatcher,
UFM_PARAM_SECURITY_LEVEL, &value);

// Set fast mode on
value = TRUE;
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_FAST_MODE,
&value);
```

Visual Basic 6.0

```
' hMatcher comes from section 2
Dim ufm_res As UFM_STATUS

' Set security level to 4
ufm_res = UFM_SetParameter(hMatcher,
UFM_PARAM_SECURITY_LEVEL, 4)

' Set fast mode on
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_FAST_MODE, 1)
```

Visual C#

```
// Matcher comes from section 2

// Set security level to 4
Matcher.SecurityLevel = 4;

// Set fast mode on
Matcher.FastMode = true;
```

Visual Basic .NET

```
' Matcher comes from section 2

' Set security level to 4
Matcher.SecurityLevel = 4

' Set fast mode on
Matcher.FastMode = true
```

java

```
// hMatcher comes from section 2
int nRes;
IntByReference value = new IntByReference;

// Set security level to 3
pValue.setValue(3);
nRes =
libMatcher.UMF_SetParameter(hMatcher, libMatcher.UMF_PARAM_SECU
```

```
RITY_LEVEL, pValue);

//Set fast mode on
pValue.setValue(1);
nRes = libMatcher.UMF_SetParameter(hMatcher,
libMatcher.UMF_PARAM_FAST_MODE, pValue)
```

4. Identify

Visual C++

```
// hMatcher comes from section 2
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], 0, MAX_TEMPLATE_SIZE);
}

// Get Template1 from scanner or image or database
// Get Template2Array from scanner or image or 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) {
    // Execute error handling codes
} else {
    if (nMatchIndex != -1) {
        // Identification succeed
    } else {
        // Identification failed
    }
}

// Free Template2Array
for (i = 0; i < MAX_TEMPLATE_NUM; i++) {
    free(Template2Array[i]);
}
```

Visual Basic 6.0

```
' hMatcher comes from section 2
Dim ufm_res As UFM_STATUS
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2Array(MAX_TEMPLATE_SIZE - 1, MAX_TEMPLATE_NUM - 1) As Byte
Dim Template2SizeArray(MAX_TEMPLATE_NUM - 1) As Long
Dim Template2Num As Long
Dim MatchIndex As Long

' Get Template1 from scanner or image or database
' Get Template2Array from scanner or image or database

' Make template pointer array to pass two dimensional
template data
Dim Template2Ptr(MAX_TEMPLATE_NUM - 1) As Long
Dim i As Long
For i = 0 To MAX_TEMPLATE_NUM - 1
    Template2Ptr(i) = VarPtr(Template2Array(0, i))
Next

' Identify Template1 from Template2Array, set timeout to 5
seconds
ufm_res = UFM_Identify(hMatcher, Template1(0),
Template1Size, Template2Ptr(0), Template2SizeArray(0),
Template2Num, 5000, nMatchIndex);

If (ufm_res <> UFM_STATUS.OK) Then
    ' Execute error handling codes
Else
    If (nMatchIndex <> -1) Then
        ' Identification succeed
    Else
        ' Identification failed
    End If
End If
```

Visual C#

```
// Matcher comes from section 2
UFM_STATUS ufm_res;
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
int Template1Size;
byte[][] Template2Array;
int[] Template2SizeArray = new int[MAX_TEMPLATE_NUM];
int Template2Num;
int nMatchIndex;
```

```

// Allocate Template2Array
Template2Array = new byte[MAX_TEMPLATE_NUM] [];
for (i = 0; i < MAX_TEMPLATE_NUM; i++) {
    Template2Array[i] = new byte[MAX_TEMPLATE_SIZE];
}

// Get Template1 from scanner or image or database
// Get Template2Array from scanner or image or database

// Identify Template1 from Template2Array, set timeout to 5
// seconds
ufm_res = Matcher.Identify(Template1, Template1Size,
Template2, Template2Size, Template2Num, 5000, out
MatchIndex);

if (ufm_res != UFM_OK) {
    // Execute error handling codes
} else {
    if (nMatchIndex != -1) {
        // Identification succeed
    } else {
        // Identification failed
    }
}

```

Visual Basic .NET

```

' hMatcher comes from section 2
Dim ufm_res As UFM_STATUS
Dim Template1 As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim Template1Size As Integer = Nothing
Dim Template2Array As Byte()()
Dim Template2SizeArray As Integer() = New
Integer(MAX_TEMPLATE_NUM) {}
Dim Template2Num As Integer
Dim MatchIndex As Integer

' Allocate Template2Array
Template2Array = New Byte(MAX_TEMPLATE_NUM)() {}
For i As Integer = 0 To MAX_TEMPLATE_NUM - 1
    Template2Array(i) = New Byte(MAX_TEMPLATE_SIZE) {}
Next

' Get Template1 from scanner or image or database
' Get Template2Array from scanner or image or database

' Identify Template1 from Template2Array, set timeout to 5
' seconds
ufm_res = Matcher.Identify(Template1, Template1Size,
Template2Array, Template2SizeArray, Template2Num, 5000,
MatchIndex)

```

```

If (ufm_res <> UFM_STATUS.OK) Then
    ' Execute error handling codes
Else
    If (MatchIndex <> -1) Then
        ' Identification succeed
    Else
        ' Identification failed
    End If
End If

```

```

java
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufm_res;
Pointer hMatcher;
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
int nTemplate1Size;

byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
int nTemplate2Num;
IntByReference bIdentifySucceed = new IntByReference();
int i;

//load library(libMatcher)

// Create hMatcher

// Get Template1

ufm_res = libMatcher.UMF_IdentifyInit(hMatcher, Template1, nTemplate1Size);
if(ufm_res == libMatcher.UMF_OK) {
    // UMF_IdentifyInit is succeeded
} else {
    // UMF_IdentifyInit is failed
    // Use UMF_GetErrorString function to show error string
}

for (i = 0; i < nTemplate2Num; i++) {
    // Get one template in DB or something, and save it to Template2 and
    nTemplate2Size

    ufm_res = libMatcher.UMF_IdentifyNext(hMatcher, Template2, nTemplate2Size,
    bIdentifySucceed);
    if(ufm_res == libMatcher.UMF_OK) {
        // UMF_IdentifyNext is succeeded
    } else {

```

```

    // UFM_IdentifyNext is failed
    // Use UFM_GetErrorString function to show error string
    // return;
}

if(bIdentifySucceed.getValue()) {
    // Identification is succeed
    break;
}
if(!bIdentifySucceed.getValue()) {
    // Identification is failed
}

```

5. Delete matcher

Visual C++

```

// hMatcher comes from section 2
UFM_STATUS ufm_res;

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

```

Visual Basic 6.0

```

' hMatcher comes from section 2
Dim ufm_res As UFM_STATUS

' Delete matcher
ufm_res = UFM_Delete(hMatcher)

```

Visual C#

```

// No explicit delete code is needed

```

Visual Basic .NET

```

' No explicit delete code is needed

```

java

```

//hMatcher comes from section 2
int nRes;

//Delete matcher
nRes= libMatcher.UFM_Delete(hMatcher);

```

Manage database

0. Required product

[Suprema BioMini SDK](#)

1. Preliminaries

Visual C++

```
// Add Suprema UFDatabase lib (lib\UFDatabase.lib) to the Project
// Add following statements in the source
#include "UFDatabase.h"

// We use 384 bytes template size in this tutorial
#define MAX_TEMPLATE_SIZE 384
```

Visual Basic 6.0

```
' Add Suprema type library (bin\Suprema.tlb) using browse button in the References
dialog (drop down the Project menu and select the References item)

Option Explicit
Option Base 0

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Long = 384
```

Visual C#

```
// Add Suprema UFDatabase library (bin\Suprema.UFDatabase.dll) using browse tap in
the Add References dialog
// Add following statements in the source
using Suprema

// We use 384 bytes template size in this tutorial
const int MAX_TEMPLATE_SIZE = 384;
```

Visual Basic .NET

```
' Add Suprema UFDatabase library (bin\Suprema.UFDatabase.dll) using browse tap in the
Add References dialog
' Add following statements in the source
Imports Suprema

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Integer = 384
```

2. Open database

Visual C++

```
UFD_STATUS ufd_res;
HUFDatabase hDatabase;

// Generate connection string
CString szConnection;
CString szDataSource("mdb_path\\UFDatabase.mdb");
szConnection.Format(TEXT("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=%s;"),
szDataSource);

// Open database
ufd_res = UFD_Open(szConnection, NULL, NULL, &hDatabase);
// Always check status return codes after running SDK functions
// Meaning of status return code can be retrieved using UFD_GetErrorString()
// In the tutorial, we omit error check codes
```

Visual Basic 6.0

```
Dim ufd_res As UFD_STATUS
Dim hDatabase As Long

' Generate connection string
Dim Connection As String
Dim DataSource As String
DataSource = "mdb_path\\UFDatabase.mdb"
Connection = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & DataSource & ";"

' Open database
ufd_res = UFD_Open(Connection, "", "", hDatabase)
' Always check status return codes after running SDK functions
' Meaning of status return code can be retrieved using UFD_GetErrorString()
' In the tutorial, we omit error check codes
```

Visual C#

```
UFS_STATUS ufd_res;
UFDatabase Database;

// Generate connection string
string Connection;
string DataSource = "mdb_path\\UFDatabase.mdb";
Connection = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & DataSource & ";";

// Open database
ufd_res = Database.Open(Connection, null, null);
// Always check status return codes after running SDK functions
// Meaning of status return code can be retrieved using UFDatabase.GetErrorString()
// In the tutorial, we omit error check codes
```

Visual Basic .NET
<pre>Dim ufd_res As UFD_STATUS Dim hDatabase As UFDatabase ' Generate connection string Dim Connection As String Dim DataSource As String DataSource = "mdb_path\UFDatabase.mdb" Connection = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & DataSource & ";" ' Open database ufd_res = Database.Open(Connection, "", "") ' Always check status return codes after running SDK functions ' Meaning of status return code can be retrieved using UFDatabase.GetErrorString() ' In the tutorial, we omit error check codes</pre>

3. Add one database entry

Visual C++
<pre>// hDatabase comes from section 2 UFD_STATUS ufd_res; CString strUserID; int nFingerIndex; unsigned char Template1[MAX_TEMPLATE_SIZE]; int Template1Size; unsigned char Template2[MAX_TEMPLATE_SIZE]; int Template2Size; CString strMemo; // Get strUserID from user // Get nFingerIndex from user // Get Template1 from scanner or image or database // Get Template2 from scanner or image or database // Get strMemo from user // Add data ufd_res = UFD_AddData(hDatabase, strUserID, nFingerIndex, Template1, Template1Size, Template2, Template2Size, strMemo);</pre>

Visual Basic 6.0
<pre>' hDatabase comes from section 2 Dim ufd_res As UFD_STATUS Dim UserID As String Dim FingerIndex As Long Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte Dim Template1Size As Long Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte</pre>

```

Dim Template2Size As Long
Dim Memo As String

' Get UserID from user
' Get FingerIndex from user
' Get Template1 from scanner or image or database
' Get Template2 from scanner or image or database
' Get Memo from user

' Add data
ufd_res = UFD_AddData(hDatabase, UserID, FingerIndex, Template1(0), Template1Size,
Template2(0), Template2Size, Memo)

```

Visual C#

```

// Database comes from section 2
UFD_STATUS ufd_res;
string UserID;
int FingerIndex;
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
int Template1Size;
byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int Template2Size;
string Memo;

// Get strUserID from user
// Get nFingerIndex from user
// Get Template1 from scanner or image or database
// Get Template2 from scanner or image or database
// Get strMemo from user

// Add data
ufd_res = Database.AddData(UserID, FingerIndex, Template1, Template1Size,
Template2, Template2Size, Memo);

```

Visual Basic .NET

```

'Database comes from section 2
Dim ufd_res As UFD_STATUS
Dim UserID As String
Dim FingerIndex As Integer
Dim Template1 As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim Template1Size As Integer = Nothing
Dim Template2 As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim Template2Size As Integer = Nothing
Dim Memo As String

' Get UserID from user
' Get FingerIndex from user
' Get Template1 from scanner or image or database
' Get Template2 from scanner or image or database

```

```
' Get Memo from user  
  
' Add data  
ufd_res = Database.AddData(UserID, FingerIndex, Template1, Template1Size,  
Template2, Template2Size, Memo)
```

4. Get first database entry

Visual C++

```
// hDatabase comes from section 2  
UFD_STATUS ufd_res;  
int nSerial;  
CString strUserID;  
int nFingerIndex;  
unsigned char Template1[MAX_TEMPLATE_SIZE];  
int Template1Size;  
unsigned char Template2[MAX_TEMPLATE_SIZE];  
int Template2Size;  
CString strMemo;  
  
' Get data  
ufd_res = UFD_GetDataByIndex(hDatabase, 0, nSerial, strUserID, nFingerIndex,  
Template1, Template1Size, Template2, Template2Size, strMemo);
```

Visual Basic 6.0

```
' hDatabase comes from section 2  
Dim ufd_res As UFD_STATUS  
Dim Serial As Long  
Dim UserID As String  
Dim FingerIndex As Long  
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte  
Dim Template1Size As Long  
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte  
Dim Template2Size As Long  
Dim Memo As String  
  
' Get data  
ufd_res = UFD_GetDataByIndex(hDatabase, 0, Serial, UserID, FingerIndex,  
Template1(0), Template1Size, Template2(0), Template2Size, Memo)
```

Visual C#

```
// Database comes from section 2  
UFD_STATUS ufd_res;  
int Serial;  
string UserID;  
int FingerIndex;  
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];  
int Template1Size;
```

```

byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int Template2Size;
string Memo;

// Add data
ufd_res = Database.GetDataByIndex(UserID, 0, out Serial, out FingerIndex, Template1,
out Template1Size, Template2, out Template2Size, out Memo);

```

Visual Basic .NET

```

'Database comes from section 2
Dim ufd_res As UFD_STATUS
Dim Serial As Integer
Dim UserID As String
Dim FingerIndex As Integer
Dim Template1 As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim Template1Size As Integer = Nothing
Dim Template2 As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim Template2Size As Integer = Nothing
Dim Memo As String

' Get data
ufd_res = Database.GetDataByIndex(UserID, 0, Serial, FingerIndex, Template1,
Template1Size, Template2, Template2Size, Memo)

```

5. Close database

Visual C++

```

// hDatabase comes from section 2
UFD_STATUS ufd_res;

// Close database
ufd_res = UFD_Close(hDatabase);

```

Visual Basic 6.0

```

'hDatabase comes from section 2
Dim ufd_res As UFD_STATUS

' Close database
ufd_res = UFD_Close(hDatabase)

```

Visual C#

```

// Database comes from section 2
UFS_STATUS ufd_res;

// Close database
ufd_res = Database.Close();

```

Suprema PC SDK 3.4.1

```
Visual Basic .NET
```

```
' Database comes from section 2  
Dim ufd_res As UFD_STATUS
```

```
' Close database  
ufd_res = Database.Close()
```

Enroll fingerprints from image

0. Required product

[Suprema Image SDK](#)

1. Preliminaries

Visual C++

```
// Add Suprema UFExtractor lib (lib\UFExtractor.lib) to the Project
// Add following statements in the source
#include "UFExtractor.h"

// We use 384 bytes template size in this tutorial
#define MAX_TEMPLATE_SIZE 384
// We assume maximum input image size to 640 x 640
#define MAX_IMAGE_WIDTH 640
#define MAX_IMAGE_HEIGHT 640
```

Visual Basic 6.0

```
' Add Suprema type library (bin\Suprema.tlb) using browse button in the References
dialog (drop down the Project menu and select the References item)

Option Explicit
Option Base 0

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Long = 384
' We assume maximum input image size to 640 x 640
Const MAX_IMAGE_WIDTH As Long = 640
Const MAX_IMAGE_HEIGHT As Long = 640
```

Visual C#

```
// Add Suprema UFExtractor library (bin\Suprema.UFExtractor.dll) using browse tap in
the Add References dialog
// Add following statements in the source
using Suprema

// We use 384 bytes template size in this tutorial
const int MAX_TEMPLATE_SIZE = 384;
// We assume maximum input image size to 640 x 640
const int MAX_IMAGE_WIDTH = 640;
const int MAX_IMAGE_HEIGHT = 640;
```

Visual Basic .NET

```
' Add Suprema UFExtractor library (bin\Suprema.UFExtractor.dll) using browse tap in the
Add References dialog
' Add following statements in the source
Imports Suprema

' We use 384 bytes template size in this tutorial
Const MAX_TEMPLATE_SIZE As Integer = 384
' We assume maximum input image size to 640 x 640
Const MAX_IMAGE_WIDTH As Integer = 640
Const MAX_IMAGE_HEIGHT As Integer = 640
```

```
java
//import Suprema java package
import com.suprema.ufe31.*;

// We use 384 bytes template size in this tutorial
//Set maximum template number to 50 (number depends on application)
public final int MAX_TEMPLATE_SIZE=384;
public final int MAX_TEMPLATE_NUM=50;

// We assume maximum input image size to 640 x 640
public final int MAX_IMAGE_WIDTH = 640;
public final int MAX_IMAGE_HEIGHT = 640;
```

2. Create extractor

Visual C++

```
UFM_STATUS ufm_res;
HUFExtractor hExtractor;

// Create extractor
ufe_res = UFE_Create(&hExtractor);
// Always check status return codes after running SDK functions
// Meaning of status return code can be retrieved using UFE_GetErrorString()
// In the tutorial, we omit error check codes
```

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS
Dim hExtractor As Long

' Create extractor
ufe_res = UFE_Create(hExtractor)
' Always check status return codes after running SDK functions
' Meaning of status return code can be retrieved using UFE_GetErrorString()
' In the tutorial, we omit error check codes
```

Visual C#

```
UFExtractor Extractor;
```

```
// Create extractor
Extractor = new Extractor();
// Always check status return codes after running SDK functions
// Meaning of status return code can be retrieved using UFExtractor.GetErrorString()
// In the tutorial, we omit error check codes
```

Visual Basic .NET

```
Dim Matcher As UFMatcher
```

```
' Create matcher
```

```
Extractor = New Extractor()
```

```
' Always check status return codes after running SDK functions
```

```
' Meaning of status return code can be retrieved using UFExtractor.GetErrorString()
```

```
' In the tutorial, we omit error check codes
```

java

```
int nRes;
```

```
Pointer hExtract =null;
```

```
PointerByReference refExtract = new PointerByReference;
```

```
UFExtractorClass libExtract=null;
```

```
//load UFExtract.dll (UFExtractorClass.class ≡ UFExtract.dll에 매플되는 wrapper
//class)
```

```
libExtractor =
```

```
(UFExtractorClass)Native.loadLibrary("UFExtract",UFExtractorClass.class);
```

```
// Create matcher
```

```
nRes = libExtractor.UFM_Create(refExtract );
```

```
hExtract= refExtractor.getValue();
```

```
// Always check status return codes after running SDK functions
```

```
// Meaning of status return code can be retrieved using UFE_GetErrorString()
```

```
// In the tutorial, we omit error check codes
```

3. Set parameters

Visual C++

```
// hExtractor comes from section 2
```

```
UFE_STATUS ufe_res;
```

```
int value;
```

```
// Set template size to 384 bytes
```

```
value = MAX_TEMPLATE_SIZE;
```

```
ufe_res = UFE_SetParameter(hExtractor, UFS_PARAM_TEMPLATE_SIZE, &value);
```

```
// Set not to detect core when extracting template
```

```
value = FALSE;
```

```
ufe_res = UFE_SetParameter(hExtractor, UFS_PARAM_DETECT_CORE, &value);
```

Visual Basic 6.0

```
' hExtractor comes from section 2
Dim ufe_res As UFE_STATUS

' Set template size to 384 bytes
ufe_res = UFE_SetParameter(hExtractor, UFS_PARAM_TEMPLATE_SIZE,
MAX_TEMPLATE_SIZE);

' Set not to detect core when extracting template
ufe_res = UFE_SetParameter(hExtractor, UFS_PARAM_DETECT_CORE, 0);
```

Visual C#

```
// Extractor comes from section 2

// Set template size to 384 bytes
Extractor.TemplateSize = MAX_TEMPLATE_SIZE;

// Set not to detect core when extracting template
Extractor.DetectCore = false;
```

Visual Basic .NET

```
' Extractor comes from section 2

' Set template size to 384 bytes
Extractor.TemplateSize = MAX_TEMPLATE_SIZE

' Set not to detect core when extracting template
Extractor.DetectCore = false
```

4. Load image and extract template

Visual C++

```
// hExtractor comes from section 2
UFE_STATUS ufe_res;
CString FileName;
unsigned char* pImage;
int nWidth;
int nHeight;
unsigned char Template[MAX_TEMPLATE_SIZE];
int TemplateSize;
int nEnrollQuality;

// Allocate image buffer
pImage = (unsigned char*)malloc(MAX_IMAGE_WIDTH * MAX_IMAGE_HEIGHT);

// Get FileName from user
```

```
// Load bitmap image
ufe_res = UFE_LoadImageFromBMPFile(FileName, pImage, &nWidth, &nHeight);

// Extract template
ufe_res = UFE_Extract(hExtractor, pImage, nWidth, nHeight, 500, Template,
&TemplateSize, &nEnrollQuality);

// Free image buffer
free(pImage);
```

Visual Basic 6.0

```
' hExtractor comes from section 2
Dim ufe_res As UFE_STATUS
Dim FileName As String
Dim Image(MAX_IMAGE_WIDTH*MAX_IMAGE_HEIGHT) As Byte
Dim Width As Long
Dim Height As Long
Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateSize As Long
Dim EnrollQuality As Long

' Get FileName from user

' Load bitmap image
ufe_res = UFE_LoadImageFromBMPFile(FileName, Image(0), Width, Height)

' Extract template
ufe_res = UFE_Extract(m_hExtractor, Image(0), Width, Height, 500, Template(0),
TemplateSize, EnrollQuality)
```

Visual C#

```
// Extractor comes from section 2
UFE_STATUS ufe_res;
string FileName;
byte[] ImageData = new byte[MAX_IMAGE_WIDTH * MAX_IMAGE_HEIGHT];
int Width;
int Height;
byte[] Template = new byte[MAX_TEMPLATE_SIZE];
int TemplateSize;
int EnrollQuality;

// Get FileName from user

// Load bitmap image
ufe_res = Extractor.LoadImageFromBMPFile(FileName, ImageData, out Width, out
Height);

// Extract template
ufe_res = Extractor.Extract(ImageData, Width, Height, 500, Template, out TemplateSize,
```

```
    out EnrollQuality);
```

Visual Basic .NET

```
' Extractor comes from section 2
Dim ufe_res As UFE_STATUS
Dim FileName As String
Dim ImageData As Byte() = New
Byte(MAX_IMAGE_WIDTH*MAX_IMAGE_HEIGHT) {}
Dim Width As Long
Dim Height As Long
Dim Template As Byte() = New Byte(MAX_TEMPLATE_SIZE) {}
Dim TemplateSize As Integer = Nothing
Dim EnrollQuality As Long

' Get FileName from user

' Load bitmap image
ufe_res = Extractor.LoadImageFromBMPFile(FileName, ImageData, Width, Height)

' Extract template
ufe_res = Extractor.Extract(ImageData, Width, Height, 500, Template, TemplateSize,
EnrollQuality)
```

5. Delete matcher

Visual C++

```
// hExtractor comes from section 2
UFE_STATUS ufe_res;

// Delete extractor
ufe_res = UFE_Delete(&hExtractor);
```

Visual Basic 6.0

```
' hMatcher comes from section 2
Dim ufe_res As UFE_STATUS

' Delete matcher
ufe_res = UFE_Delete(hExtractor)
```

Visual C#

```
// No explicit delete code is needed
```

Visual Basic .NET

```
' No explicit delete code is needed
```

Java Installation guide

0. Required product

You need the following packages installed

Suprema PC SDK ,Java SDK 1.4 or later,JNA library(jna.jar that following location
"<suprema pc sdk installed path>/bin")

1. Install Java SDK

you must install Java SDK 1.4. or later version for JNA

please see <http://java.sun.com> site.

copy the jna.jar to "<java sdk installed path>/lib"

2. Set Classpath

After java sdk installation,you must set classpath as following example(windows)

"CLASSPATH=.;<java sdk installed path>/bin;<java sdk installed path>/lib;<java sdk installed path>/lib/jna.jar;<suprema pc sdk installed path>/bin;"

3. Build and run sample

you can build and run the JNA demo application in the following location.

<installed folder>/samples/java/JNA/javac demoUFE33JavaJNA.java

<installed folder>/samples/java/JNA/java demoUFE33JavaJNA

you can build and run the JNI demo application in the following location.

<installed folder>/samples/java/JNI/javac demoUFE33JavaJNI.java

<installed folder>/samples/java/JNI/java demoUFE33JavaJNI

Sample Demo Source guide

UFE30_Demo

Required product

[Suprema BioMini SDK](#)

Task

- Makes a connection with fingerprint scanner
- Enrolls captured fingerprint
- Verification
- Identification
- Saves an image or template file
- Shows fingerprint image at image frame component
- Option settings in capturing process and matching process

UFE30_ImageDemo

Required product

[Suprema Image SDK](#)

Task

- Loads fingerprint image file
- Shows fingerprint image file at the image frame component
- Extracts a template from fingerprint image
- Verifies two fingerprint images

UFE30_DatabaseDemo

Required product

[Suprema BioMini SDK](#)

Task

- Makes a connection with fingerprint scanner
- Loads MS Access database file(mdb)
- Enrolls captured fingerprint and insert data into database
- Verification
- Identification
- Deletes user data from database
- Updates user information into database
- Updates user fingerprint template into database

UFE30_EnrollDemo

Required product

[Suprema BioMini Enroll SDK](#)

Task

- Makes a connection with fingerprint scanner
- Enrolls captured fingerprint
- Saves an image or template file
- Shows fingerprint image at image frame component
- Option settings in capturing process and matching process

UFE30_MultiScannerDemo

Required product

[Suprema BioMini SDK](#)

Task

- Makes connection with multiple fingerprint scanner
- Enrolls captured fingerprint
- Identification
- Shows fingerprint image at image frame component

Fake fingerprint detection

0. Required product

[Suprema BioMini SDK](#) and [BioMini Plus Scanner](#)

1. Set fake finger detection option

Visual C++

```
HUFScanner hScanner;
UFS_STATUS ufs_res;
int nDetectFake;

// Get first scanner handle
ufs_res = UFS_GetScannerHandle(0, &hScanner);

// Fake fingerprint detection level
nDetectFake = 2;

// Set fake fingerprint detection option
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_DETECT_FAKE,
&nDetectFake);
```

About LFD level - level 0 : don't use LFD function

level 1 : weak detection of fake fingerprint but real finger accept rate is high.

level 2 : normal detection of fake fingerprint. (recommended level)

level 3 : strong detection of fake fingerprint but real finger accept rate is low.

2. Use fake finger detection

Visual C++

```
HUFScanner hScanner;
UFS_STATUS ufs_res;

// Get first scanner handle (0 means first scanner)
ufs_res = UFS_GetScannerHandle(0, &hScanner);

// Use fake finger detection with fingerprint capturing. Fake finger detection is activated
// with UFS_CaptureSingleImage function only
ufs_res = UFS_CaptureSingleImage(hScanner);
```

Chapter 4. Sample

Suprema PC SDK supplies various type of sample applications as follows,

Samples	Remarks
<u>UFR30_EnrollDemo</u>	Provides the basic usage about managing scanners and executing enrollment
<u>UFR30_Demo</u>	Provides the basic usage about managing scanners and executing enrollment, verification and identification
<u>UFE30_DatabaseDemo</u>	Provides the demo about managing database
<u>UFE30_ImageDemo</u>	Provides the demo about extracting templates from images and matching two templates
<u>UFE30_MultiScannerDemo</u>	Provides the demo about using multiple scanners simultaneously

Before running sample applications

- For sample applications using scanners: connect scanner to PC, and install scanner driver (install\drivers) appropriate to each scanner model.
- Confirm Suprema PC SDK module DLLs are exist in the same folder with sample application EXEs or module DLLs are installed in the windows system folder.
- Confirm the license file (UFLicense.dat) exist in the same folder with Suprema PC SDK module DLLs.
- Java SDK is using JNA that provides simplified access to native library methods. please see following site. <https://jna.dev.java.net>

UFE30_Demo

UFE30_Demo provides the basic usage about managing scanners and executing enrollment, verification and identification. This program uses [UFScanner](#) and [UFMatcher](#) modules.

Required products

Products	Remarks
Suprema BioMini SDK	No restriction

Available languages

Languages	Locations
Visual C++ 6.0	samples\VS60\UFE30_DemoVC60
Visual Basic 6.0	samples\VS60\UFE30_DemoVB60
Visual C#	samples\VS80\UFE30_DemoCS
Visual Basic .NET	samples\VS80\UFE30_DemoVBNET
java	samples\java\UFE30_DemoJava

UFE30_Demo_Usage

Executable File Location

- bin\UFE30_DemoCS60.exe
- bin\UFE30_DemoVBNET.exe
- bin\UFE30_DemoCS.exe
- Java version
- Source code of all demo application

Required products

[Suprema BioMini SDK](#)

Picture of the Demo sample applications

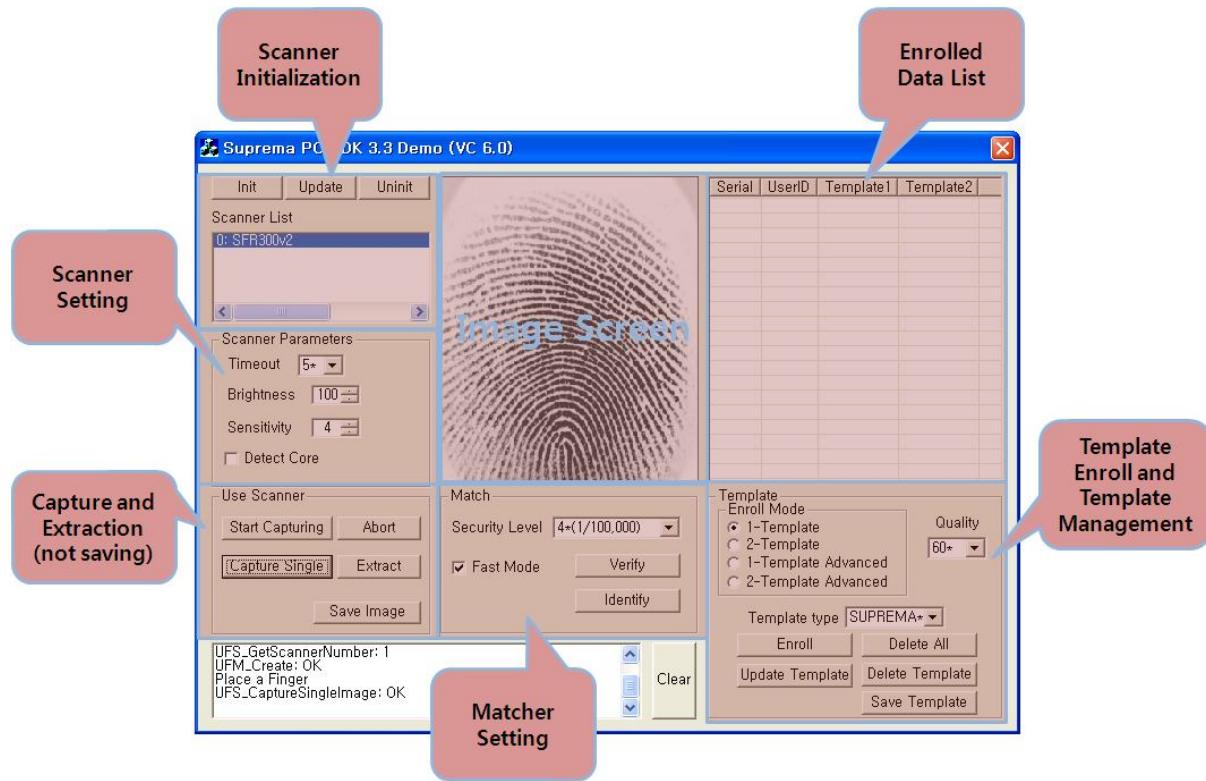
User Interface Components

Suprema PC SDK Demo sample demonstrates how to use Suprema PC SDK roughly. Please use this sample as a reference for making your own program. PC SDK demo provides following methods:

- Initialize Scanner - A scanner should be initialized for using all functions about the scanner
- Enrolls fingerprint - A fingerprint can be enrolled by using BioMini scanner
- Verification - A fingerprint can be verified against enrolled fingerprint
- Identification - A fingerprint can be identified against every enrolled fingerprint
- Saves data - Saves a template file or a fingerprint image to BMP format

Every UFE30_Demo application is written by different languages from Visual C++ to JAVA. But they are made of same functions and same interface design.

The picture bellow shows the PC SDK Demo sample main window and description about bunch of interface components :



The table below shows the PC SDK Demo's simple functionality and links to each detailed usage page :

Scanner initialize	
The items about scanner initialization. Using these components, to makes a connection or disconnection with fingerprint scanner and checking the connection state	
Init button - Initializes UFScanner module	Source
Code	
Update button - Checks the condition of connection state with scanner	Source Code
Uninit button - Uninitializes UFScanner module	Source Code
ScannerList list box - List of the scanners	
Scanner options	
Option items for capturing process	
Timeout combo box - Countdown value for the 'capture' functions	
Brightness spin Edit - Sets the brightness of connected scanner	Source Code
Sensitivity spin Edit - Sets the sensitivity of connected scanner	Source Code
Detect Core check box - If this check box is checked program captures an image only when find the 'core' in the fingerprint	Source Code

User Scanner	
Items for capturing a fingerprint image and extracts a template from captured image	
Start Capturing button - Captures an image with callback function	Source Code
Abort button - Cancels the start capturing	Source Code
Caputre Single button - Captures a fingerprint image	Source Code
Extract button - Extracts a template from captured image (not saving the template)	Source Code

Template	
Items for enrolling fingerprint templates and management the saved templates	
Template type combo box - Sets the template type	Source Code
Quality Combo box - Quality threshold of template quality	
Enroll button - Enrolls one or two fingerprint	Source Code
Delete All button - Deletes all enrolled data	
Update Template button - Update the template of selected data	
Delete Template button - Deletes selected data	

Match	
Items for verification and identification	
Security Level Combo box - Sets the level of security. Higher value more exactly detects the correct fingerprint but false reject rate will be increased	Source Code
Fast Mode Check box - If this check box is checked the matching speed will be increased	Source Code
Verify button - Verifies input fingerprint whether matched or not compared with selected fingerprint	Source Code
Identify button - Identifies input fingerprint from enrolled fingerprints	Source Code

Save Template - Saves an image Template
Save Image - Saves an image of bmp format

Scanner Initialization

Description

To capture a fingerprint image, one or more scanner has to be initialized with your program. This page is the source code about initialization task that is connected with 'Init' button in 'demo' samples.

Using functions

[UFS_Init](#), [UFS_GetErrorString](#), [UFS_SetScannerCallback](#), [UFS_GetScannerNumber](#),
[UFM_Create](#), [UFM_GetErrorString](#)

Source Code

The table bellow shows the 'Init button' source code and description in 'Demo' samples:

VS60
<pre>void CUFE30_DemoDlg::OnInit() { ///////////////// // Initialize scanner module and get scanner list ///////////////// <ufs_res takes UFSanner function's return value> UFS_STATUS ufs_res; <number of connected scanners> int nScannerNumber; BeginWaitCursor(); <scanner initialization function see UFS_Init to check the return value. This function returns true value if the initialization success, false otherwise> ufs_res = UFS_Init(); EndWaitCursor(); if(ufs_res == UFS_OK) { <updates a message to the message list box component> AddMessage("UFS_Init: OK\r\n"); } else { <gets an error message string according to ufs_res which is a return value of UFS function> UFS_GetErrorString(ufs_res, m_strError); AddMessage("UFS_Init: %s\r\n", m_strError); } }</pre>

```

}

<sets a callback function for checking status of scanner>
ufs_res = UFS_SetScannerCallback(ScannerProc, (void*)this);
if(ufs_res == UFS_OK) {
    AddMessage("UFS_SetScannerCallback: OK\r\n");
} else {
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_SetScannerCallback: %s\r\n", m_strError);
    return;
}
<gets the number of scanners>
ufs_res = UFS_GetScannerNumber(&nScannerNumber);
if(ufs_res == UFS_OK) {
    AddMessage("UFS_GetScannerNumber: %d\r\n", nScannerNumber);
} else {
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_GetScannerNumber: %s\r\n", m_strError);
    return;
}
UpdateScannerList();
Invalidate(TRUE);

///////////////////////////////
// Create one matcher
/////////////////////////////
<ufm_res takes UFMatcher function's return value>
UFM_STATUS ufm_res;
<creates a matcher which is responsible for matching process>
ufm_res = UFM_Create(&m_hMatcher);
if(ufm_res == UFM_OK) {
    AddMessage("UFM_Create: OK\r\n");
} else {
    <gets an error message string according to ufm_res which is a return value of UFM
    functions>
    UFM_GetErrorString(ufm_res, m_strError);
    AddMessage("UFM_Create: %s\r\n", m_strError);
    return;
}
GetMatcherSettings(m_hMatcher);
/////////////////////////////
}

```

VB60

```

Private Sub btnInit_Click()
'=====
' Initialize scanners
'=====

    <ufs_res takes UFSanner function's return value>
    Dim ufs_res As UFS_STATUS
        <number of connected scanners>
    Dim ScannerNumber As Long

    Screen.MousePointer = vbHourglass
        <scanner initialization function
        see UFS\_Init to check the return values. This function returns true value if the
        initialization success, false otherwise>
    ufs_res = UFS_Init()
    Screen.MousePointer = vbDefault
    If(ufs_res = UFS_STATUS.OK) Then
        <updates a message to the message list in main window>
        AddMessage ("UFS_Init: OK" & vbCrLf)
    Else
        <gets an error message string according to ufs_res which is a return value of UFS
        function>
        UFS_GetErrorString ufs_res, m_strError
        AddMessage ("UFS_Init: " & m_strError & vbCrLf)
        Exit Sub
    End If
    <sets a callback function for checking status of scanner>
    ufs_res = UFS_SetScannerCallback(AddressOf ScannerProc, 0)
    If(ufs_res = UFS_STATUS.OK) Then
        AddMessage ("UFS_SetScannerCallback: OK" & vbCrLf)
    Else
        UFS_GetErrorString ufs_res, m_strError
        AddMessage ("UFS_SetScannerCallback: " & m_strError & vbCrLf)
        Exit Sub
    End If
    <gets the number of scanners>
    ufs_res = UFS_GetScannerNumber(ScannerNumber)
    If(ufs_res = UFS_STATUS.OK) Then
        AddMessage ("UFS_GetScannerNumber: " & ScannerNumber & vbCrLf)
    Else
        UFS_GetErrorString ufs_res, m_strError
        AddMessage ("UFS_GetScannerNumber: " & m_strError & vbCrLf)
        Exit Sub
    End If

    UpdateScannerList
'=====

```

```

'=====
'====='
' Create matcher
'====='

    <ufm_res takes UFMATCHER function's return value>
Dim ufm_res As UFM_STATUS
    <creates a matcher which is responsible for matching process>
ufm_res = UFM_Create(m_hMatcher)
If (ufm_res = UFM_STATUS.OK) Then
    AddMessage ("UFM_Create: OK" & vbCrLf)
Else
    <gets an error message string according to ufm_res which is a return value of UFM
function>
    UFM_GetErrorString ufm_res, m_strError
    AddMessage ("UFM_Create: " & m_strError & vbCrLf)
    Exit Sub
End If

GetMatcherSettings (m_hMatcher)
'====='

End Sub

```

C#

```

private void btnInit_Click(object sender, EventArgs e) {
//=====
//=====//
// Initialize scanners
//=====//
//=====//
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <number of connected scanners>
    int nScannerNumber;
    Cursor.Current = Cursors.WaitCursor;
    <scanner initialization function
    see UFS\_Init to check the return values. This function returns true value if the
    initialization success, false otherwise>
    ufs_res = m_ScannerManager.Init();
    Cursor.Current = this.Cursor;
    if(ufs_res == UFS_STATUS.OK) {
        tbxMessage.AppendText("UFSanner Init: OK\r\n");
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFSanner.GetErrorString(ufs_res, out m_strError);
    }
}

```

```
    tbxMessage.AppendText("UFSscanner Init: " + m_strError + "\r\n");
    return;
}
<sets a callback function for checking status of scanner>
m_ScannerManager.ScannerEvent += new UFS_SCANNER_PROC(ScannerEvent);
<gets the number of scanners>
nScannerNumber = m_ScannerManager.Scanners.Count;
tbxMessage.AppendText("UFSscanner GetScannerNumber: " + nScannerNumber +
"\r\n");
<updates a message to the message list component in main window>
UpdateScannerList();
//=====
=====//
//=====
=====//
// Create matcher
//=====
=====//
<creates a matcher which is responsible for matching process>
m_Matcher = new UFMatcher();
GetMatcherSettings(m_Matcher);
//=====
=====//
}
}
```

Update Scanner

Description

The OnUpdate(VS60) function. See the source code blow. This function refreshes the connected scanner list when an additional scanner will be connected or current scanner will be removed.

Using functions

[UFS_Update](#), [UFS_GetErrorString](#)

Source Code

VS60

```
void CUFE30_DemoDlg::OnUpdate()
{
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    BeginWaitCursor();
    <refreshes the list of scanners. See UFS\_Update to check return value>
    ufs_res = UFS_Update();
    EndWaitCursor();
    if(ufs_res == UFS_OK) {
        AddMessage("UFS_Update: OK\r\n");
        UpdateScannerList();
        Invalidate(TRUE);
    } else {
        <gets an error message string according to ufs_res which is a return value of UFS
        functions>
        UFS_GetErrorString(ufs_res, m_strError);
        AddMessage("UFS_Update: %s\r\n", m_strError);
    }
}
```

VB60

```
Private Sub btnUpdate_Click()
    <ufs_res takes UFSanner function's return value>
    Dim ufs_res As UFM_STATUS

    Screen.MousePointer = vbHourglass
    <refreshes the list of scanners. See UFS\_Update to check return value>
    ufs_res = UFS_Update()
    Screen.MousePointer = vbDefault
```

```
If (ufs_res = UFS_STATUS.OK) Then
    AddMessage ("UFS_Update: OK" & vbCrLf)
    UpdateScannerList
Else
    <gets an error message string according to ufs_res which is a return value of UFS
functions>
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_Update: " & m_strError & vbCrLf)
End If
End Sub
```

C#

```
private void btnUpdate_Click(object sender, EventArgs e) {
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    Cursor.Current = Cursors.WaitCursor;
    <refreshes the list of scanners. See UFS\_Update to check return value>
    ufs_res = m_ScannerManager.Update();
    Cursor.Current = this.Cursor;
    if (ufs_res == UFS_STATUS.OK) {
        tbxMessage.AppendText("UFScanner Update: OK\r\n");
        UpdateScannerList();
    } else {
        <gets an error message string according to ufs_res which is a return value of
        the UFS functions>
        UFScanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFScanner Update: " + m_strError + "\r\n");
    }
}
```

Scanner Uninitialization

Description

Demo sample's 'uninit' button is set to calling the OnUninit(VS60) function. To use the scanner properly and prevent unexpected errors, scanner resources should be released when they are unused. Use the UFS_Uninit function to release the resources that were allocated at initialization.

Using functions

[UFS_Uninit](#), [UFS_GetErrorString](#), [UFM_Delete](#), [UFM_GetErrorString](#)

Source Code

```
VS60
void CUFE30_DemoDlg::OnUninit()
{
    //////////////////////////////////////////////////////////////////
    // Uninit scanner module
    //////////////////////////////////////////////////////////////////
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    BeginWaitCursor();
    <uninitializes a scanner, use this to release resources which are related to the
    scanner. See UFS\_Uninit to check the return values>
    ufs_res = UFS_Uninit();
    EndWaitCursor();
    if(ufs_res == UFS_OK) {
        AddMessage("UFS_Uninit: OK\r\n");
        m_ctlScannerList.ResetContent();
        Invalidate(TRUE);
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFS_GetErrorString(ufs_res, m_strError);
        AddMessage("UFS_Uninit: %s\r\n", m_strError);
    }
    //////////////////////////////////////////////////////////////////
    // Delete matcher
    //////////////////////////////////////////////////////////////////
    if(m_hMatcher != NULL) {
        UFM_STATUS ufm_res;
```

```

<releases matcher's resources. See UFM\_Delete>
ufm_res = UFM_Delete(m_hMatcher);
if(ufm_res == UFM_OK) {
    AddMessage("UFM_Delete: OK\r\n");
} else {
    <gets an error message string according to ufm_res which is a return
    value of UFM function>
    UFM_GetErrorString(ufm_res, m_strError);
    AddMessage("UFM_Delete: %s\r\n", m_strError);
}
m_hMatcher = NULL;
}
//=====
}

```

VB60

```

Private Sub btnUninit_Click()
'=====
' Uninit scanners
'=====

    <ufs_res takes UFSanner function's return value>
Dim ufs_res As UFM_STATUS

Screen.MousePointer = vbHourglass
    <uninitializes a scanner, use this to release the resources which are related to the
    scanner. See UFS\_Uninit to check the return value>
ufs_res = UFS_Uninit()
Screen.MousePointer = vbDefault
If (ufs_res = UFS_STATUS.OK) Then
    AddMessage ("UFS_Uninit: OK" & vbCrLf)
    lbScannerList.Clear
Else
    <gets an error message string according to ufs_res which is a return value of UFS
function>
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_Uninit: " & m_strError & vbCrLf)
End If
'=====

' Delete matcher
'=====

If (m_hMatcher <> 0) Then
    <ufm_res takes UFMatcher function's return value>

```

```

Dim ufm_res As UFM_STATUS
    <releases matcher's resources. See UFM\_Delete to check the return value>
    ufm_res = UFM_Delete(m_hMatcher)
    If (ufm_res = UFM_STATUS.OK) Then
        AddMessage ("UFM_Delete: OK" & vbCrLf)
    Else
        <gets an error message string according to ufm_res which is a return value
        of UFM function>
        UFM_GetErrorString ufm_res, m_strError
        AddMessage ("UFM_Delete: " & m_strError & vbCrLf)
    End If
    m_hMatcher = 0
End If
'=====
=====
'
End Sub

```

C#

```

private void btnUninit_Click(object sender, EventArgs e) {
    //=====
    //=====
    // Uninit Scanners
    //=====
    //=====
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;

    Cursor.Current = Cursors.WaitCursor;
    <uninitializes a scanner, use this to release the resources which are related to the
    scanner. See UFS\_Uninit to check the return value>
    ufs_res = m_ScannerManager.Uninit();
    Cursor.Current = this.Cursor;
    if (ufs_res == UFS_STATUS.OK) {
        tbxMessage.AppendText("UFScanner Uninit: OK\r\n");
        m_ScannerManager.ScannerEvent -= ScannerEvent;
        lbScannerList.Items.Clear();
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFSanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFScanner Uninit: " + m_strError + "\r\n");
    }
    pbImageFrame.Image = null;
    //=====
    //=====
}

```

Capturing Option

Description

The options relate to the capturing process.

The 'timeout' value is a countdown value applied to UFS_StartCapturing function and UFS_CaptureSingle function. The UFS_SetParameter function sets this value with UFS_PARAM_TIMEOUT parameter. If this value is set to '0', the scanner waits a finger input eternally, in UFS_CaptureSingle function. In the case of UFS_StartCapturing function, the scanner starts infinite capturing loop.

The 'Brightness' is a brightness value applied to captured fingerprint image. The UFS_SetParameter function sets this value with UFS_PARAM_BRIGHTNESS parameter.

The 'Sensitivity' value influences to determine whether the scanner captures an image or not. If the sensitivity value is high, scanner will capture an image even if the fingerprint area is small or the clarity of image is bad.

If you set the 'Detect core', scanner will capture a fingerprint image only when the 'core' feature is existed in the fingerprint.

Using functions

[UFS_GetParameter](#), [UFS_GetScannerHandle](#) in GetCurrentScannerHandle

Source Code

VS60

```
void CUFE30_DemoDlg::GetCurrentScannerSettings()
{
    HUFScanner hScanner;
    int value;
    <gets a scanner handle>
    if (!GetCurrentScannerHandle(&hScanner)) {
        return;
    }
    // Unit of timeout is millisecond
    <gets a timeout value>
    UFS_GetParameter(hScanner, UFS_PARAM_TIMEOUT, &value);
    m_nTimeout = value / 1000;
    ((CComboBox*)GetDlgItem(IDC_TIMEOUT))->SetCurSel(m_nTimeout);
    <gets a brightness value>
    UFS_GetParameter(hScanner, UFS_PARAM_BRIGHTNESS, &value);
```

```

    m_nBrightness = value;
    ((CSpinButtonCtrl*)GetDlgItem(IDC_BRIGHTNESS_SPIN))->SetRange(0,
255);
<gets a sensitivity value>
UFS_GetParameter(hScanner, UFS_PARAM_SENSITIVITY, &value);
    m_nSensitivity = value;
    ((CSpinButtonCtrl*)GetDlgItem(IDC_SENSITIVITY_SPIN))->SetRange(0, 7);
<gets a detect core value>
UFS_GetParameter(hScanner, UFS_PARAM_DETECT_CORE, &value);
    m_bDetectCore = value;
    UpdateData(FALSE);
}

```

VB60

```

Private Sub GetCurrentScannerSettings()
    <ufs_res takes every UFSanner function's return value>
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim value As Long
<gets a scanner handle>
If (GetCurrentScannerHandle(hScanner) = False) Then
    Exit Sub
End If

' Unit of timeout is millisecond
    <gets a timeout value>
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM.TIMEOUT, value)
cbTimeout.ListIndex = value / 1000
<gets a brightness value>
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM.BRIGHTNESS, value)
tbBrightness.text = value
<gets a sensitivity value>
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM.SENSITIVITY, value)
tbSensitivity.text = value
<gets a detect core value>
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM.DETECT_CORE, value)
cbDetectCore.value = value
End Sub

```

C#

```
private void GetCurrentScannerSettings()
{
    <gets a scanner handle>
    UFScanner Scanner;
    if (!GetGetCurrentScanner(out Scanner)) {
        return;
    }

    // Unit of timeout is millisecond
    <gets a timeout value>
    cbTimeout.SelectedIndex = Scanner.Timeout / 1000;
    nudBrightness.Minimum = 0;
    nudBrightness.Maximum = 255;
    <gets a brightness value>
    nudBrightness.Value = Scanner.Brightness;
    nudSensitivity.Minimum = 0;
    nudSensitivity.Maximum = 7;
    <gets a sensitivity value>
    nudSensitivity.Value = Scanner.Sensitivity;
    <gets a detect core value>
    cbDetectCore.Checked = Scanner.DetectCore;
}
```

Start Capturing

Description

Demo sample's 'start capturing' button is set to calling the OnStartCapturing(VS60) function. This function captures a fingerprint image from scanner by using UFS_StartCapturing function. the capturing process will be operated during 'timeout' value.

Example of timeout setting

```
value = m_nTimeout * 1000;
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT, &value);
```

Using functions

[UFS_StartCapturing](#), [UFS_GetErrorString](#), [UFS_GetScannerHandle](#) in
GetCurrentScannerHandle

Source Code

VS60

```
void CUFE30_DemoDlg::OnStartCapturing()
{
    HUFScanner hScanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <gets a scanner handle>
    if (!GetCurrentScannerHandle(&hScanner)) {
        return;
    }
    // Test code: for checking fps
    m_frame_num = 0;
    <captures an image from the scanner after TIMEOUT(sec) countdown.
    If you set this value to 0, the scanner keep on capturing until 'Abort' button is
    pressed>
    ufs_res = UFS_StartCapturing(hScanner, CaptureProc, (void*)this);
    if (ufs_res == UFS_OK) {
        AddMessage("UFS_StartCapturing: OK\r\n");
        Invalidate(FALSE);
    } else {
        <gets an error message string according to ufs_res which is a return value of UFS
        function>
```

```
        UFS_GetErrorString(ufs_res, m_strError);
        AddMessage("UFS_StartCapturing: %s\r\n", m_strError);
    }
}
```

VB60

```
Private Sub btnStartCapturing_Click()
    <ufs_res takes UFSanner function's return value>
    Dim ufs_res As UFM_STATUS
    Dim hScanner As Long
    <gets a scanner handle>
    If (GetCurrentScannerHandle(hScanner) <> True) Then
        Exit Sub
    End If

    <captures an image from the scanner after TIMEOUT(sec) countdown.
    If you set this value to 0, the scanner keep on capturing until 'Abort' button is
    pressed>
    ufs_res = UFS_StartCapturing(hScanner, AddressOf CaptureProc, 0)
    If (ufs_res = UFS_STATUS.OK) Then
        AddMessage ("UFS_StartCapturing: OK" & vbCrLf)
        pbImageFrame.Refresh
    Else
        <gets an error message string according to ufs_res which is a return value of UFS
        function>
        UFS_GetErrorString ufs_res, m_strError
        AddMessage ("UFS_StartCapturing: " & m_strError & vbCrLf)
    End If
End Sub
```

C#

```

private void btnStartCapturing_Click(object sender, EventArgs e) {
    UFSscanner Scanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <gets a scanner handle>
    if (!GetGetCurrentScanner(out Scanner)) {
        return;
    }
    <sets a callback function for taking an image frame for realtime>
    Scanner.CaptureEvent += new UFS_CAPTURE_PROC(CaptureEvent);
    <captures an image from the scanner after TIMEOUT(sec) countdown.
    If you set this value to 0, the scanner keep on capturing until 'Abort' button is
    pressed>
    ufs_res = Scanner.StartCapturing();
    if (ufs_res == UFS_STATUS.OK) {
        tbxMessage.AppendText("UFSscanner StartCapturing: OK\r\n");
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFSscanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFSscanner StartCapturing: " + m_strError + "\r\n");
    }
}

```

Abort Capturing

Description

Demo sample's 'abort' button is set to calling the OnAbortCapturing(VS60) function. This function provides a cancellation function applied to image capturing process by calling UFS_AbortCapturing. The UFS_AbortCapturing is available after a capturing starts.

Check

The UFS_AbortCapturing function should be must called in a different thread from the capturing thread. The UFS_StartCapturing makes a specific thread that operates image capturing, internally. Thus UFS_AbortCapturing could be used with UFS_StartCapturing function directly. But the UFS_CaptureSingleImage does not make a new independent thread. In case that the UFS_AbortCapturing and the UFS_CaptureSingleImage are used together, it should be necessary to make a new thread.

Using functions

[UFS_AbortCapturing](#), [UFS_GetErrorString](#), [UFS_GetScannerHandle](#) in
GetCurrentScannerHandle

Source Code

VS60

```
void CUFE30_DemoDlg::OnAbortCapturing()
{
    HUFScanner hScanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <gets a scanner handle>
    if (!GetCurrentScannerHandle(&hScanner)) {
        return;
    }
    <aborts capturing>
    ufs_res = UFS_AbortCapturing(hScanner);
    if (ufs_res == UFS_OK) {
        AddMessage("UFS_AbortCapturing: OK\r\n");
    } else {
        <gets an error message string according to ufs_res which is a return value of UFS
        function>
        UFS_GetErrorString(ufs_res, m_strError);
        AddMessage("UFS_AbortCapturing: %s\r\n", m_strError);
    }
}
```

```
// Test code: for checking fps
AddMessage("Current frame num: %d\r\n", m_frame_num);
}
```

VB60

```
Private Sub btnAbortCapturing_Click()
    <ufs_res takes UFSanner function's return value>
    Dim ufs_res As UFM_STATUS
    Dim hScanner As Long
    <gets a scanner handle>
    If (GetCurrentScannerHandle(hScanner) <> True) Then
        Exit Sub
    End If
    <aborts capturing>
    ufs_res = UFS_AbortCapturing(hScanner)
    If (ufs_res = UFS_STATUS.OK) Then
        AddMessage ("UFS_AbortCapturing: OK" & vbCrLf)
    Else
        <gets an error message string according to ufs_res which is a return value of UFS
        function>
        UFS_GetErrorString ufs_res, m_strError
        AddMessage ("UFS_AbortCapturing: " & m_strError & vbCrLf)
    End If
End Sub
```

C#

```
private void btnAbortCapturing_Click(object sender, EventArgs e) {
    UFScanner Scanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <gets a scanner handle>
    if (!GetCurrentScanner(out Scanner)) {
        return;
    }
    <aborts capturing>
    ufs_res = Scanner.AbortCapturing();
    if (ufs_res == UFS_STATUS.OK) {
        tbxMessage.AppendText("UFScanner AbortCapturing: OK\r\n");
    } else {
        <gets an error message string according to ufs_res which is a return value of UFS
        function>
        UFScanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFScanner AbortCapturing: " + m_strError +
        "\r\n");
    }
}
```

Capture Single

Description

Demo sample's OnCaptureSingle function(VS60) captures a fingerprint image from the connected scanner by using UFS_CaptureSingleImage function. When this function is called, the scanner will be standby. And it will capture an image when user puts one's finger on the prism.

Using functions

[UFS_CaptureSingleImage](#), [UFS_GetErrorString](#), [UFS_GetScannerHandle](#) in
GetCurrentScannerHandle

Source Code

VS60

```
void CUFE30_DemoDlg::OnCaptureSingle()
{
    HUFScanner hScanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <gets a scanner handle>
    if (!GetCurrentScannerHandle(&hScanner)) {
        return;
    }
    BeginWaitCursor();
    <captures a fingerprint image>
    ufs_res = UFS_CaptureSingleImage(hScanner);
    EndWaitCursor();
    if (ufs_res == UFS_OK) {
        AddMessage("UFS_CaptureSingleImage: OK\r\n");
        Invalidate(FALSE);
    } else {
        <gets an error message string according to ufs_res which is a return value of UFS
        function>
        UFS_GetErrorString(ufs_res, m_strError);
        AddMessage("UFS_CaptureSingleImage: %s\r\n", m_strError);
    }
}
```

VB60

```
Private Sub btnCaptureSingle_Click()
    <ufs_res takes UFSanner function's return value>
```

```

Dim ufs_res As UFM_STATUS
Dim hScanner As Long
    <gets a scanner handle>
If (GetCurrentScannerHandle(hScanner) <> True) Then
    Exit Sub
End If
AddMessage ("Place your finger" & vbCrLf)

Screen.MousePointer = vbHourglass
    <captures a fingerprint image>
ufs_res = UFS_CaptureSingleImage(hScanner)
Screen.MousePointer = vbDefault
If (ufs_res = UFS_STATUS.OK) Then
    AddMessage ("UFS_CaptureSingleImage: OK" & vbCrLf)
    pbImageFrame.Refresh
Else
    <gets an error message string according to ufs_res which is a return value of UFS
function>
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_CaptureSingleImage: " & m_strError & vbCrLf)
End If
End Sub

```

C#

```

private void btnCaptureSingle_Click(object sender, EventArgs e) {
    UFScanner Scanner;
    <ufs_res takes UFScanner function's return value>
    UFS_STATUS ufs_res;
    <gets a scanner handle>
    if (!GetGetCurrentScanner(out Scanner)) {
        return;
    }
    Cursor.Current = Cursors.WaitCursor;
    <captures a fingerprint image>
    ufs_res = Scanner.CaptureSingleImage();
    Cursor.Current = this.Cursor;
    if (ufs_res == UFS_STATUS.OK) {
        tbxMessage.AppendText("UFScanner CaptureSingleImage: OK\r\n");
        DrawCapturedImage(Scanner);
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFScanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFScanner CaptureSingleImage: " + m_strError +
        "\r\n");
    }
}

```

Template Extraction

Description

To verify or identify a fingerprint template, it should be extracted in accordance with the data format of fingerprint features.

Demo sample's 'Extract' button is set to calling OnExtract(VS60). This function sets the template type, and then extracts a template from captured fingerprint image using UFS_Extract function.

Using functions

[UFS_SetParameter](#), [UFS_SetTemplateType](#), [UFS_Extract](#),
[UFS_GetErrorString](#), [UFS_GetScannerHandle](#) in GetCurrentScannerHandle

Source Code

VS60
<pre>void CUFE30_DemoDlg::OnExtract() { UpdateData(TRUE); HUFScanner hScanner; <ufs_res takes UFSanner function's return value> UFS_STATUS ufs_res; int value; char szType[20]; <gets a the scanner handle> if (!GetCurrentScannerHandle(&hScanner)) { return; } value = MAX_TEMPLATE_SIZE; <sets the template size> UFS_SetParameter(hScanner, UFS_PARAM_TEMPLATE_SIZE, &value); value = m_bDetectCore; <sets the detect core> UFS_SetParameter(hScanner, UFS_PARAM_DETECT_CORE, &value); <m_nType is a dialog member variable to set the type of template> switch (m_nType) { case 0: <sets the template type to SUPREMA> UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_SUPREMA); AddMessage("Current scanner,UFS_SetTemplateType: SUPREMA TYPE \r\n"); sprintf(szType,"%s","suprema type"); } }</pre>

```

break;
case 1:
<sets the template type to ISO>
UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_ISO19794_2);
AddMessage("Current scanner,UFS_SetTemplateType: ISO_19794_2 TYPE
\r\n");
sprintf(szType,"%s","iso_19794_2 type");
break;
case 2:
<sets the template type to ANSI>
UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_ANSI378);
AddMessage("Current scanner,UFS_SetTemplateType: ANSI378 TYPE \r\n");
sprintf(szType,"%s","ansi378 type");
break;
}
<makes a template variable, the default size is 384
bytes(=MAX_TEMPLATE_SIZE)>
unsignd char Template[MAX_TEMPLATE_SIZE];
int TemplateSize;
int nEnrollQuality;
BeginWaitCursor();
<extracts a template from captured fingerprint image. The extracted template data
will be saved in the Template variable>
ufs_res = UFS_Extract(hScanner, Template, &TemplateSize, &nEnrollQuality);
EndWaitCursor();
if(ufs_res == UFS_OK) {
    AddMessage("UFS_Extract: OK %s \r\n",szType);
    Invalidate(FALSE);
} else {
<gets ab error message string according to ufs_res which is a return value of UFS
function>
    UFS_GetErrorString(ufs_res, m_strerror);
    AddMessage("UFS_Extract:%s, %s\r\n",szType, m_strerror);
}
}
}

```

VB60

```

Private Sub btnExtract_Click()
Dim hScanner As Long
<ufs_res takes UFSanner function's return value>
Dim ufs_res As UFS_STATUS
Dim value As Long
<gets a scanner handle>
If (GetCurrentScannerHandle(hScanner) <> True) Then
    Exit Sub
End If
value = MAX_TEMPLATE_SIZE
<sets the template size>
UFS_SetParameter hScanner, UFS_PARAM TEMPLATE_SIZE, value

```

```

value = cbDetectCore.value
    <sets the detect core>
UFS_SetParameter hScanner, UFS_PARAM.DETECT_CORE, value
    <cbType.ListIndex is a dialog member variable to select the type of template>
If(cbType.ListIndex = 0) Then
    <sets the template type to SUPREMA>
    UFS_SetTemplateType hScanner,
UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_SUPREMA
ElseIf(cbType.ListIndex = 1) Then
    <sets the template type to ISO>
    UFS_SetTemplateType hScanner,
UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ISO19794_2
ElseIf(cbType.ListIndex = 2) Then
    <sets the template type to ANSI>
    UFS_SetTemplateType hScanner,
UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ANSI378
End If

<makes a template variable, the default size is 384 bytes(=MAX_TEMPLATE_SIZE)>
Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateSize As Long
Dim EnrollQuality As Long

Screen.MousePointer = vbHourglass
    <extracts a template from captured fingerprint image. The extracted template data will
be saved in the Template variable>
ufs_res = UFS_Extract(hScanner, Template(0), TemplateSize, EnrollQuality)
Screen.MousePointer = vbDefault
If(ufs_res = UFS_STATUS.OK) Then
    AddMessage ("UFS_Extract: OK" & vbCrLf)
    pbImageFrame.Refresh
Else
    <gets ab error message string according to ufs_res which is a return value of UFS
function>
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_Extract: " & m_strError & vbCrLf)
End If
End Sub

```

C#

```

private void btnExtract_Click(object sender, EventArgs e) {
    UFSscanner Scanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <gets a scanner handle>
    if (!GetGetCurrentScanner(out Scanner)) {
        return;
    }
    <sets the template size>
    Scanner.TemplateSize = MAX_TEMPLATE_SIZE;
    <sets the detect core>
    Scanner.DetectCore = cbDetectCore.Checked;
    <makes a template variable, the default size is 384 bytes(=MAX_TEMPLATE_SIZE)>
    byte[] Template = new byte[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    int EnrollQuality;
    Cursor.Current = Cursors.WaitCursor;
    <extracts a template from captured fingerprint image. The extracted template data
    will be saved in the Template variable>
    ufs_res = Scanner.Extract(Template, out TemplateSize, out EnrollQuality);
    Cursor.Current = this.Cursor;
    if (ufs_res == UFS_STATUS.OK) {
        tbxMessage.AppendText("UFSscanner Extract: OK\r\n");
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFSscanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFSscanner Extract: " + m_strError + "\r\n");
    }
}

```

Enrollment

Description

Enrolling a fingerprint is extracting a template from finger and saving the template. Demo sample's OnEnroll(VS60) function captures a fingerprint image after setting the template type. And calls the UFS_Extract function to extract a template from captured fingerprint image. The extracted template will be saved in a specific template array which is a parameter of the UFS_Extract function.

Using functions

[UFS_SetTemplateType](#), [UFS_ClearCaptureImageBuffer](#), [UFS_CaptureSingleImage](#),
[UFS_Extract](#), [UFS_GetErrorString](#), [UFS_GetScannerHandle](#) in GetCurrentScannerHandle

Source Code

VS60
<pre>void CUFE30_DemoDlg::OnEnroll() { UpdateData(TRUE); HUFScanner hScanner; <ufs_res takes UFSanner function's return value> UFS_STATUS ufs_res; int nEnrollQuality; char szType[20]; <gets a scanner handle> if (!GetCurrentScannerHandle(&hScanner)) { return; } switch (m_nType) { case 0: <sets the template type to SUPREMA> UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_SUPREMA); AddMessage("Current scanner,UFS_SetTemplateType: SUPREMA TYPE \r\n"); sprintf(szType,"%s","suprema type"); break; case 1: <sets the template type to ISO> UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_ISO19794_2); AddMessage("Current scanner,UFS_SetTemplateType: ISO_19794_2 TYPE \r\n"); } }</pre>

```

sprintf(szType,"%s","iso_19794_2 type");
break;
case 2:
<sets the template type to ANSI378>
UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_ANSI378);
AddMessage("Current scanner,UFS_SetTemplateType: ANSI378 TYPE \r\n");
sprintf(szType,"%s","ansi378 type");
break;
}
<dialog for advanced enroll>
CUFE30_EnrollDlg dlg;
<dialog for getting user ID>
CUFE30_UserInfoDlg udlg;
AddMessage("Input user data\r\n");
if(udlg.DoModal() != IDOK) {
    AddMessage("User data input is cancelled by user\r\n");
    return;
}
<enroll with Non-Advanced-Extraction>
if(m_nEnrollMode == 0 || m_nEnrollMode == 3) {
    UFS_ClearCaptureImageBuffer(hScanner);
    AddMessage("Place a finger\r\n");

    while (TRUE) {
        ufs_res = UFS_CaptureSingleImage(hScanner);
        if(ufs_res != UFS_OK) {
            UFS_GetErrorString(ufs_res, m_strError);
            AddMessage("UFS_CaptureSingleImage: %s\r\n", m_strError);
            return;
        }
        Invalidate(FALSE);

        if(m_template_num+1 == MAX_TEMPLATE_NUM) {
            AddMessage("Template memory is full\r\n");
        }

        <extract first template>
        if(template_enrolled == 0)
            ufs_res = UFS_Extract(hScanner,
                m_template1[m_template_num],
                &m_templateSize1[m_template_num], &nEnrollQuality);
        <extract second template>
        else
            ufs_res = UFS_Extract(hScanner,
                m_template2[m_template_num],
                &m_templateSize2[m_template_num], &nEnrollQuality);
        if(ufs_res == UFS_OK) {
            AddMessage("UFS_Extract: OK %s\r\n",szType);
        }
    }
}

```

```

<template quality check (can be modified from Quality control in
main dialog)>
if(nEnrollQuality < m_quality) {
    AddMessage("Template Quality is too low\r\n");
}
else {
    strcpy(m_userid[m_template_num], udlg.m_strUserID);
    template_enrolled++;
    AddMessage("Enrollment success (No.%d) [Q:%d]\r\n",
    m_template_num, nEnrollQuality);
    <if enroll mode is set to take 1 template>
    if(m_nEnrollMode == 0) {
        m_template_num++;
        UpdateFingerDataList();
        break;
    <if enroll mode is set to take 2 template>
    } else if (m_nEnrollMode == 3 && template_enrolled ==
    2){
        m_template_num++;
        UpdateFingerDataList();
        break;
    } else {
        AddMessage("Remove finger\r\n");
        while(1) {
            ufs_res = UFS_IsFingerOn(hScanner,
            &fingeron);
            if(fingeron == 0) {
                AddMessage("Place a finger\r\n");
                break;
            }
        }
    }
} else {
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_Extract:%s, %s\r\n",szType,m_strError);
}
}
}

<enroll with Advanced-Extraction>
else {
    <initialize template buffer>
    for( i = 0; i < m_nEnrollMode; i++)
    {
        m_enrolltemplate[i] = (unsigned
        char*)malloc(MAX_TEMPLATE_SIZE);
        m_enrolltemplateSize[i] = 0;
    }
}

```

```

dlg.m_hScanner = hScanner;
dlg.m_strUserID = udlg.m_strUserID;
dlg.m_output_num = m_nEnrollMode;
dlg.m_EnrollTemplate = m_enrolltemplate;
dlg.m_EnrollTemplateSize = m_enrolltemplateSize;
dlg.m_quality = m_quality;

<call advanced enroll dialog (refer to sample source code to check advanced
enroll usage)>
if(dlg.DoModal() != IDOK) {
    AddMessage("Fingerprint enrollment is cancelled by user\r\n");
    for( i = 0; i < m_nEnrollMode; i++) {
        free(m_enrolltemplate[i]);
    }
    return;
}
<if advanced enroll is finished successfully>
if(m_enrolltemplateSize[0] != 0) {
    if(m_template_num+1 == MAX_TEMPLATE_NUM) {
        AddMessage("Template memory is full\r\n");
    } else {
        <if enroll mode is set to take 1 template>
        if(m_nEnrollMode == 1) {
            memcpy(m_template1[m_template_num],
m_enrolltemplate[0], m_enrolltemplateSize[0]);
            m_templateSize1[m_template_num] =
m_enrolltemplateSize[0];
            strcpy(m_userid[m_template_num], udlg.m_strUserID);
        }
        <if enroll mode is set to take 2 template>
        else {
            memcpy(m_template1[m_template_num],
m_enrolltemplate[0], m_enrolltemplateSize[0]);
            m_templateSize1[m_template_num] =
m_enrolltemplateSize[0];
            memcpy(m_template2[m_template_num],
m_enrolltemplate[1], m_enrolltemplateSize[1]);
            m_templateSize2[m_template_num] =
m_enrolltemplateSize[1];
            strcpy(m_userid[m_template_num], udlg.m_strUserID);
        }
        AddMessage("Enrollment is succeed (No.%d)\r\n",
m_template_num);
        m_template_num++;
    }
    UpdateFingerDataList();
} else {
    AddMessage("Enrollment is failed\r\n");
}

```

```
<restore the timeout option to main dialog setting>
value = m_nTimeout * 1000;
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT, &value);
if(ufs_res != UFS_OK) {
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_SetParameter(UFS_PARAM_TIMEOUT): %s\r\n",
               m_strError);
}
<release template buffer>
for( i = 0; i < m_nEnrollMode; i++) {
    free(m_enrolltemplate[i]);
}
}
```

VB60

```

Private Sub btnEnroll_Click()
    <ufs_res takes UFSanner function's return value>
    Dim ufs_res As UFS_STATUS
    Dim hScanner As Long
    Dim EnrollQuality As Long
    Dim EnrollMode As Long
    Dim TemplateSize As Long
    Dim template_enrolled As Long
    Dim fingeron As Long
    <gets a scanner handle>
    If (GetCurrentScannerHandle(hScanner) <> True) Then
        Exit Sub
    End If
    AddMessage ("Place your finger" & vbCrLf)

    If (cbType.ListIndex = 0) Then
        <sets a template type to the SUPREMA>
        UFS_SetTemplateType hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_SUPREMA
    ElseIf (cbType.ListIndex = 1) Then
        <sets a template type to the ISO>
        UFS_SetTemplateType hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ISO19794_2
    ElseIf (cbType.ListIndex = 2) Then
        <sets a template type to the ANSI>
        UFS_SetTemplateType hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ANSI378
    End If

    If (obtnAdEnroll1 = True) Then
        EnrollMode = 1
    Else
        EnrollMode = 2
    End If
End Sub

```

```

End If

template_enrolled = 0

AddMessage ("Input user data" & vbCrLf)
UFE30_User_Info.SetAdd
UFE30_User_Info.Show vbModal
<dialog for getting user ID>
If (Not UFE30_User_Info.DialogResult) Then
    AddMessage ("User data input is cancelled by user" & vbCrLf)
    Exit Sub
End If

<enroll with Non-Advanced-Extraction>
If (obtnEnroll = True Or obtnEnroll2 = True) Then
    UFS_ClearCaptureImageBuffer (hScanner)
    AddMessage ("Place a finger" & vbCrLf)

Do
    ufs_res = UFS_CaptureSingleImage(hScanner)
    If (ufs_res <> UFS_STATUS.OK) Then
        UFS_GetErrorString ufs_res, m_strError
        AddMessage ("UFS_CaptureSingleImage: " & m_strError & vbCrLf)
        Exit Sub
    End If

    If (m_template_num + 1 = MAX_TEMPLATE_NUM) Then
        AddMessage ("Template memory is full" & vbCrLf)
        Exit Sub
    End If

    pbImageFrame.Refresh
    If (template_enrolled = 0) Then
        <extract first template>
        ufs_res = UFS_Extract(hScanner, m_template1(0, m_template_num),
        m_templateSize1(m_template_num), EnrollQuality)
    Else
        <extract second template>
        ufs_res = UFS_Extract(hScanner, m_template2(0, m_template_num),
        m_templateSize2(m_template_num), EnrollQuality)
    End If

    If ufs_res = UFS_STATUS.OK Then
        <template quality check (can be modified from Quality control in main dialog)>
        If (EnrollQuality < m_quality) Then
            AddMessage ("Template Quality is too low" & vbCrLf)
        Else
            m UserID(m_template_num) = UFE30_User_Info.txtUserID
            template_enrolled = template_enrolled + 1
    End If
End If

```

```

    AddMessage ("Enrollment success (No." & m_template_num & ") [Q:" &
EnrollQuality & "]" & vbCrLf)
    <if enroll mode is set to take 1 template>
    If (obtnEnroll = True) Then
        m_template_num = m_template_num + 1
        UpdateFingerDataList
        Exit Do
    <if enroll mode is set to take 2 template>
    ElseIf (obtnEnroll2 = True And template_enrolled = 2) Then
        m_template_num = m_template_num + 1
        UpdateFingerDataList
        Exit Do
    Else
        AddMessage ("Remove finger" & vbCrLf)
        Do
            ufs_res = UFS_IsFingerOn(hScanner, fingeron)
            If (fingeron = 0) Then
                AddMessage ("Place a finger" & vbCrLf)
                Exit Do
            End If
            Loop While 1
        End If
    End If
    Else
        UFS_GetErrorString ufs_res, m_strError
        AddMessage ("UFS_Extract: " & m_strError & vbCrLf)
        Exit Sub
    End If
    Loop While 1
    <enroll with Advanced-Extraction>
Else
    <initialize template buffer>
    For i = 0 To EnrollMode - 1
        Call CopyMemory(ByVal VarPtr(m_enrolltemplate(0, i)), 0,
MAX_TEMPLATE_SIZE - 1)
        m_enrolltemplateSize(i) = 0
    Next

    UFE30_Enroll.m_hScanner = hScanner
    UFE30_Enroll.txtUserID = UFE30_User_Info.txtUserID
    UFE30_Enroll.OutputNum = EnrollMode
    UFE30_Enroll.Quality = m_quality
    <call advanced enroll dialog (refer to sample source code to check advanced enroll
usage)>
    UFE30_Enroll.Show vbModal
    If (Not UFE30_Enroll.DialogResult) Then
        AddMessage ("Fingerprint enrollment is cancelled by user" & vbCrLf)
        Exit Sub
    End If

```

```

<take the templates from advanced enroll dialog>
For i = 0 To EnrollMode - 1
    UFE30_Enroll.GetOutputTemplate ByVal VarPtr(m_enrolltemplate(0, i)),
    m_enrolltemplateSize(i), i
    Next
    <if advanced enroll is finished successfully>
    If (m_enrolltemplateSize(0) <> 0) Then
        If (m_template_num + 1 = MAX_TEMPLATE_NUM) Then
            AddMessage ("Template memory is full" & vbCrLf)
        Else
            <if enroll mode is set to take 1 template>
            If (obtnAdEnroll1 = True) Then
                Call CopyMemory(ById VarPtr(m_template1(0, m_template_num)), ByVal
                VarPtr(m_enrolltemplate(0, 0)), m_enrolltemplateSize(0))
                m_templateSize1(m_template_num) = m_enrolltemplateSize(0)
                m_UserID(m_template_num) = UFE30_User_Info.txtUserID
            <if enroll mode is set to take 2 template>
            Else
                Call CopyMemory(ById VarPtr(m_template1(0, m_template_num)), ByVal
                VarPtr(m_enrolltemplate(0, 0)), m_enrolltemplateSize(0))
                m_templateSize1(m_template_num) = m_enrolltemplateSize(0)

                Call CopyMemory(ById VarPtr(m_template2(0, m_template_num)), ByVal
                VarPtr(m_enrolltemplate(0, 1)), m_enrolltemplateSize(1))
                m_templateSize2(m_template_num) = m_enrolltemplateSize(1)

                m_UserID(m_template_num) = UFE30_User_Info.txtUserID
            End If
            AddMessage ("Enrollment is succeed (No." & m_template_num & ")" &
            vbCrLf)
            m_template_num = m_template_num + 1
        End If
        UpdateFingerDataList
    Else
        AddMessage ("Enrollment is failed" & vbCrLf)
    End If
    <restore the timeout option to main dialog setting>
    value = cbTimeout.ListIndex * 1000
    ufs_res = UFS_SetParameter(hScanner, UFS_PARAM.TIMEOUT, value)
    If (ufs_res <> UFS_STATUS.OK) Then
        ufs_res = UFS_GetErrorString(ufs_res, m_strError)
        AddMessage ("UFS_SetParameter(UFS_PARAM_TIMEOUT): " & m_strError &
        vbCrLf)
    End If
End If
End Sub

```

C#

```

private void btnEnroll_Click(object sender, EventArgs e) {
    UFSscanner Scanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    int EnrollQuality;
    int EnrollMode;
    int template_enrolled = 0;
    bool fingeron;
    <gets a scanner handle>
    if (!GetGetCurrentScanner(out Scanner)) {
        return;
    }
    switch (this.cbScanTemplateType.SelectedIndex) {
    case 0:
        Scanner.nTemplateType = 2001;
        break;
    case 1:
        Scanner.nTemplateType = 2002;
        break;
    case 2:
        Scanner.nTemplateType = 2003;
        break;
    }
    EnrollMode = rbtnOneTemplateAdvanced.Checked ? 1 : 2;
    <dialog for getting user ID>
    UFE30_UserInfo udlg = new UFE30_UserInfo();

    tbxMessage.AppendText("Input user data\r\n");
    if (udlg.ShowDialog(this) != DialogResult.OK) {
        tbxMessage.AppendText("User data input is cancelled by user\r\n");
        return;
    }
    <enroll with Non-Advanced-Extraction>
    if (rbtnOneTemplateNormal.Checked == true || rbtnOneTemplateNormal2.Checked == true) {
        Scanner.ClearCaptureImageBuffer();
        tbxMessage.AppendText("Place Finger\r\n");
        while (true) {
            ufs_res = Scanner.CaptureSingleImage();
            if (ufs_res != UFS_STATUS.OK) {
                UFSscanner.GetErrorString(ufs_res, out m_strError);
                tbxMessage.AppendText("UFSscanner CaptureSingleImage: " + m_strError +
                    "\r\n");
                return;
            }
            DrawCapturedImage(Scanner);
        }
        if (m_template_num + 1 == MAX_TEMPLATE_NUM) {
    }
}

```

```

        tbxMessage.AppendText("Template memory is full\r\n");
    }

    if(template_enrolled == 0)
        <extract first template>
        ufs_res = Scanner.Extract(m_template1[m_template_num], out
m_template_size1[m_template_num], out EnrollQuality);
    else
        <extract second template>
        ufs_res = Scanner.Extract(m_template2[m_template_num], out
m_template_size2[m_template_num], out EnrollQuality);
    if(ufs_res == UFS_STATUS.OK) {
        <template quality check (can be modified from Quality control in main dialog)>
        if(EnrollQuality < m_quality) {
            tbxMessage.AppendText("Too low quality [Q:" + EnrollQuality + "]\r\n");
        }
        else {
            m UserID[m_template_num] = udlg.UserID;
            template_enrolled++;
            tbxMessage.AppendText("Enrollment success (No." + m_template_num + ")"
[Q:" + EnrollQuality + "]\r\n");
            <if enroll mode is set to take 1 template>
            if(rbtnOneTemplateNormal.Checked == true) {
                m_template_num++;
                UpdateFingerDataList();
                break;
            }
            <if enroll mode is set to take 2 template>
            else if(rbtnOneTemplateNormal2.Checked == true && template_enrolled
== 2) {
                m_template_num++;
                UpdateFingerDataList();
                break;
            }
            else {
                tbxMessage.AppendText("Remove finger\r\n");
                while(true) {
                    fingeron = Scanner.IsFingerOn;
                    if(fingeron == false) {
                        tbxMessage.AppendText("Place a finger\r\n");
                        break;
                    }
                }
            }
        }
    }
    else {
        UFSscanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFSscanner Extract: " + m_strError + "\r\n");
    }
}

```

```

        }
    }
}

<enroll with Advanced-Extraction>
else {
    UFE30_Enroll dlg = new UFE30_Enroll();
    dlg.hScanner = Scanner;
    dlg.UserID = udlg.UserID;
    dlg.OutputNum = EnrollMode;
    dlg.Quality = m_quality;
    <call advanced enroll dialog (refer to sample source code to check advanced enroll usage)>
    if(dlg.ShowDialog(this) != DialogResult.OK) {
        tbxMessage.AppendText("Fingerprint enrollment is cancelled by user\r\n");
        return;
    }
    <if advanced enroll is finished successfully>
    if(dlg.EnrollTemplateOutputSize[0] != 0) {
        if(m_template_num+1 == MAX_TEMPLATE_NUM) {
            tbxMessage.AppendText("Template memory is full\r\n");
        } else {
            <if enroll mode is set to take 1 template>
            if(EnrollMode == 1) {
                System.Array.Copy(dlg.EnrollTemplateOutput[0], 0,
m_template1[m_template_num], 0, dlg.EnrollTemplateOutputSize[0]);
                m_template_size1[m_template_num] = dlg.EnrollTemplateOutputSize[0];
                m UserID[m_template_num] = udlg.UserID;
            }
            <if enroll mode is set to take 2 template>
            else {
                System.Array.Copy(dlg.EnrollTemplateOutput[0], 0,
m_template1[m_template_num], 0, dlg.EnrollTemplateOutputSize[0]);
                m_template_size1[m_template_num] = dlg.EnrollTemplateOutputSize[0];
                System.Array.Copy(dlg.EnrollTemplateOutput[1], 0,
m_template2[m_template_num], 0, dlg.EnrollTemplateOutputSize[1]);
                m_template_size2[m_template_num] = dlg.EnrollTemplateOutputSize[1];
                m UserID[m_template_num] = udlg.UserID;
            }
            tbxMessage.AppendText("Enrollment is succeed (No." + m_template_num +
")\r\n");
            m_template_num++;
        }
        UpdateFingerDataList();
    } else {
        tbxMessage.AppendText("Enrollment is failed\r\n");
    }
    <restore the timeout option to main dialog setting>
    Scanner.Timeout = cbTimeout.SelectedIndex * 1000;
}

```

{ }

Matching Option

Description

The options for matching.

Security Level - If this option is set to high value, matcher will find correct fingerprint as well as possible. But the possibility of finding success rate will be lower.

Template Type - Template type for Matching and Enrollment process should be identical.

Fast Mode - By using the 'fast mode', the matching speed will increase.

Using functions

[UFM_SetParameter](#), [UFM_GetErrorString](#)

Source Code

VS60

```
void CUFE30_DemoDlg::OnSelchangeSecurityLevel()
{
    <ufs_res takes UFSanner function's return value>
    UFM_STATUS ufm_res;
    int value;
    UpdateData(TRUE);
    // Security level ranges from 1 to 7
    value = m_nSecurityLevel + 1;
    <sets the security level>
    ufm_res = UFM_SetParameter(m_hMatcher, UFM_PARAM_SECURITY_LEVEL,
        &value);
    if(ufm_res != UFM_OK) {
        <gets an error message string according to ufm_res which is a return value of UFM
        function>
        UFM_GetErrorString(ufm_res, m_strError);
        AddMessage("UFM_SetParameter(UFM_PARAM_SECURITY_LEVEL): %s\
r\n", m_strError);
    }
}

void CUFE30_DemoDlg::OnFastMode()
{
    UFM_STATUS ufm_res;
```

```

int value;
UpdateData(TRUE);
value = m_bFastMode;
<sets the fast mode>
ufm_res = UFM_SetParameter(m_hMatcher, UFM_PARAM_FAST_MODE,
&value);
if(ufm_res != UFM_OK) {
    UFM_GetErrorString(ufm_res, m_strError);
    AddMessage("UFM_SetParameter(UMF_PARAM_FAST_MODE): %s\r\n",
    m_strError);
}
}

```

VB60

```

Private Sub cbSecurityLevel_Click()
    <ufm_res takes UFMatcher function's return value>
    Dim ufm_res As UFM_STATUS
    <sets the security level>
    ' Security level ranges from 1 to 7
    ufm_res = UFM_SetParameter(m_hMatcher, UFM_PARAM.SECURITY_LEVEL,
    cbSecurityLevel.ListIndex + 1)
    If(ufm_res <> UFM_STATUS.OK) Then
        UFM_GetErrorString ufm_res, m_strError
        AddMessage ("UFM_SetParameter(UMF_PARAM.SECURITY_LEVEL): " &
    m_strError & vbCrLf)
    End If
End Sub

Private Sub cbFastMode_Click()
    <ufs_res takes UFSanner function's return value>
    Dim ufm_res As UFM_STATUS
    <sets the fast mode>
    ufm_res = UFM_SetParameter(m_hMatcher, UFM_PARAM.FAST_MODE,
    cbFastMode.value)
    If(ufm_res <> UFM_STATUS.OK) Then
        <gets an error message string according to ufm_res which is a return value of UFM
        function>
        UFM_GetErrorString ufm_res, m_strError
        AddMessage ("UFM_SetParameter(UMF_PARAM.FAST_MODE): " & m_strError
    & vbCrLf)
    End If
End Sub

```

C#

```
private void cbSecurityLevel_SelectedIndexChanged(object sender, EventArgs e) {
    if(m_Matcher != null) {
        // Security level ranges from 1 to 7
        <sets the security level>
        m_Matcher.SecurityLevel = cbSecurityLevel.SelectedIndex + 1;
    }
}

private void cbFastMode_CheckedChanged(object sender, EventArgs e) {
    if(m_Matcher != null) {
        <sets the fast mode>
        m_Matcher.FastMode = cbFastMode.Checked;
    }
}
```

Verification

Description

Two fingerprints can be verified whether they are matched or not.

Demo sample's 'Verify' button is set to calling the OnVerify(VS60) function. This function captures a fingerprint image and extracts a template from the image. And calls the UFM_Verify function to execute 1:1 matching using extracted template and another selected template by setting at Enroll ID combobox.

Using functions

[UFS_SetTemplateType](#), [UFM_SetTemplateType](#), [UFS_ClearCaptureImageBuffer](#),
[UFS_GetErrorString](#), [UFS_Extract](#), [UFM_Verify](#), [UFM_GetErrorString](#),
[UFS_GetScannerHandle](#) in GetCurrentScannerHandle

Source Code

VS60

```
void CUFE30_DemoDlg::OnVerify()
{
    UpdateData(TRUE);
    HUFScanner hScanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <ufm_res takes UFMatcher function's return value>
    UFM_STATUS ufm_res;
    unsigned char Template[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    int nEnrollQuality;
    int bVerifySucceed;
    char szType[20];
    char szMatchingType[20];
    int nSelected;

    if(m_nSelectID == -1) {
        AddMessage("Select Enroll ID\r\n");
        return;
    }
    <gets a scanner handle>
    if(!GetCurrentScannerHandle(&hScanner)) {
        return;
    }
```

```

nSelected = m_ctlFingerDataList.GetSelectionMark();
if(nSelected == -1) {
    AddMessage("Select data\r\n");
    return;
}

<sets template type for extraction>
switch (m_nType) {
    case 0:
        UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_SUPREMA);
        AddMessage("Current scanner,UFS_SetTemplateType: SUPREMA TYPE
\r\n");
        sprintf(szType,"%s","suprema type");
        break;
    case 1:
        UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_ISO19794_2);
        AddMessage("Current scanner,UFS_SetTemplateType: ISO_19794_2 TYPE
\r\n");
        sprintf(szType,"%s","iso_19794_2 type");
        break;
    case 2:
        UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_ANSI378);
        AddMessage("Current scanner,UFS_SetTemplateType: ANSI378 TYPE \r\n");
        sprintf(szType,"%s","ansi378 type");
        break;
}
<sets template type for verification>
switch (m_nMatchingType) {
    case 0:
        UFM_SetTemplateType(m_hMatcher,
        UFM_TEMPLATE_TYPE_SUPREMA);
        AddMessage("Matching type,UFM_SetTemplateType: SUPREMA TYPE
\r\n");
        sprintf(szMatchingType,"%s","suprema type");
        break;
    case 1:
        UFM_SetTemplateType(m_hMatcher,
        UFM_TEMPLATE_TYPE_ISO19794_2);
        AddMessage("Matching type,UFM_SetTemplateType: ISO_19794_2 TYPE
\r\n");
        sprintf(szMatchingType,"%s","iso_19794_2 type");
        break;
    case 2:
        UFM_SetTemplateType(m_hMatcher, UFM_TEMPLATE_TYPE_ANSI378);
        AddMessage("Matching type,UFM_SetTemplateType: ANSI378 TYPE \r\n");
        sprintf(szMatchingType,"%s","ansi378 type");
        break;
}
UFS_ClearCaptureImageBuffer(hScanner);

```

```

AddMessage("Place a finger\r\n");
<captures a fingerprint image>
ufs_res = UFS_CaptureSingleImage(hScanner);
if (ufs_res != UFS_OK) {
<gets an error message string according to ufs_res which is a return value of UFM function>
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_CaptureSingleImage: %s\r\n", m_strError);
    return;
}
<extracts a template from captured fingerprint image>
ufs_res = UFS_Extract(hScanner, Template, &TemplateSize, &nEnrollQuality);
if (ufs_res == UFS_OK) {
    Invalidate(FALSE);
} else {
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_Extract:%s, %s\r\n", szType, m_strError);
    return;
}
<verifies first template is matched>
ufm_res = UFM_Verify(m_hMatcher, Template, TemplateSize,
m_template1[nSelected], m_templateSize1[nSelected], &bVerifySucceed);
if (ufm_res != UFM_OK) {
    UFM_GetErrorString(ufm_res, m_strError);
    AddMessage("UFM_Verify:%s, %s\r\n", szMatchingType, m_strError);
    return;
}
if (bVerifySucceed) {
    AddMessage("Verification succeed,%s (ID.%s)\r\n", szMatchingType,
    m_userid[nSelected]);
} else {
    if(m_templateSize2[nSelected] != 0) {
        <verifies second template is matched>
        ufm_res = UFM_Verify(m_hMatcher, Template, TemplateSize,
        m_template2[nSelected], m_templateSize2[nSelected],
        &bVerifySucceed);
        if (ufm_res != UFM_OK) {
            UFM_GetErrorString(ufm_res, m_strError);
            AddMessage("UFM_Verify:%s, %s\r\n", szMatchingType,
            m_strError);
            return;
        }
        if (bVerifySucceed) {
            AddMessage("Verification succeed,%s
(ID.%s)\r\n", szMatchingType, m_userid[nSelected]);
        } else {
            AddMessage("Verification failed,%s\r\n", szMatchingType);
        }
    } else {
    }
}

```

```

        AddMessage("Verification failed,%s\r\n",szMatchingType);
    }
}
}
```

VB60

```

Private Sub btnVerify_Click()
    Dim hScanner As Long
        <ufs_res takes UFSanner function's return value>
    Dim ufs_res As UFS_STATUS
        <ufm_res takes UFMatcher function's return value>
    Dim ufm_res As UFM_STATUS
    Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
    Dim TemplateSize As Long
    Dim EnrollQuality As Long
    Dim VerifySucceed As Long
    Dim SelectID As Integer
    If (Not lvFingerDataList.SelectedItem.Selected) Then
        AddMessage ("Select data" & vbCrLf)
        Exit Sub
    Else
        SelectID = Val(lvFingerDataList.SelectedItem.text)
    End If

    If (cbID.ListIndex = -1) Then
        AddMessage ("Select Enroll ID" & vbCrLf)
        Exit Sub
    End If

    <gets a scanner handle>
    If (GetCurrentScannerHandle(hScanner) <> True) Then
        Exit Sub
    End If

    <sets template type for extraction>
    If (cbType.ListIndex = 0) Then
        UFS_SetTemplateType hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_SUPREMA
    ElseIf (cbType.ListIndex = 1) Then
        UFS_SetTemplateType hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ISO19794_2
    ElseIf (cbType.ListIndex = 2) Then
        UFS_SetTemplateType hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ANSI378
    End If

    UFS_ClearCaptureImageBuffer (hScanner)
    AddMessage ("Place your finger" & vbCrLf)
```

```

<captures a fingerprint image>
ufs_res = UFS_CaptureSingleImage(hScanner)
If(ufs_res <> UFS_STATUS.OK) Then
    <gets an message string according to ufs_res which is a return value of UFS
function>
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_CaptureSingleImage: " & m_strError & vbCrLf)
    Exit Sub
End If

<extracts a template from captured fingerprint image>
ufs_res = UFS_Extract(hScanner, Template(0), TemplateSize, EnrollQuality)
If ufs_res = UFS_STATUS.OK Then
    pbImageFrame.Refresh
Else
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_Extract: " & m_strError & vbCrLf)
    Exit Sub
End If

<sets template type for verification>
If(cbMatchType.ListIndex = 0) Then
    UFM_SetTemplateType m_hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_SUPREMA
ElseIf(cbType.ListIndex = 1) Then
    UFM_SetTemplateType m_hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_ISO19794_2
ElseIf(cbType.ListIndex = 2) Then
    UFM_SetTemplateType m_hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_ANSI378
End If

<verifies first template is matched>
ufm_res = UFM_Verify(m_hMatcher, Template(0), TemplateSize, m_template1(0,
SelectID), m_templateSize1(SelectID), VerifySucceed)
If ufm_res <> UFM_STATUS.OK Then
    UFM_GetErrorString ufm_res, m_strError
    AddMessage ("UFM_Verify: " & m_strError & vbCrLf)
    Exit Sub
End If

If(VerifySucceed) Then
    AddMessage ("Verification succeed (No." & (SelectID + 1) & ")" & vbCrLf)
Else
    If(m_templateSize2(SelectID) <> 0) Then
        <verifies second template is matched>
        ufm_res = UFM_Verify(m_hMatcher, Template(0), TemplateSize, m_template2(0,
SelectID), m_templateSize2(SelectID), VerifySucceed)
        If(ufs_res <> UFS_STATUS.OK) Then

```

```

    UFM_GetErrorString ufm_res, m_strError
    AddMessage ("UFMatcher Verify: " & m_strError & vbCrLf)
End If

If (VerifySucceed) Then
    AddMessage ("Verification succeed (No." & (SelectID + 1) & ")" & vbCrLf)
Else
    AddMessage ("Verification failed" & vbCrLf)
End If
Else
    AddMessage ("Verification failed" & vbCrLf)
End If
End If
End Sub

```

C#

```

private void btnVerify_Click(object sender, EventArgs e) {
    UFSscanner Scanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <ufm_res takes UFMatcher function's return value>
    UFM_STATUS ufm_res;
    byte[] Template = new byte[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    int EnrollQuality;
    bool VerifySucceed;
    int Serial;
    if (lvFingerDataList.SelectedItems.Count == 0) {
        tbxMessage.AppendText("Select data\r\n");
        return;
    }
    else {
        Serial =
        Convert.ToInt32(lvFingerDataList.SelectedItems[0].SubItems[FINGERDATA_COL_SERIAL].Text);
    }
    <gets a scanner handle>
    if (!GetGetCurrentScanner(out Scanner)) {
        return;
    }
    Scanner.ClearCaptureImageBuffer();
    tbxMessage.AppendText("Place Finger\r\n");
    <captures a fingerprint image>
    ufs_res = Scanner.CaptureSingleImage();
    if (ufs_res != UFS_STATUS.OK) {
        <gets an error message string according to ufs_res which is a return value of UFS
        function>
        UFSscanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFSscanner CaptureSingleImage: " + m_strError +

```

```

    "\r\n");
    return;
}

switch (this.cbScanTemplateType.SelectedIndex)
{
    case 0:
        Scanner.nTemplateType = 2001;
        break;
    case 1:
        Scanner.nTemplateType = 2002;
        break;
    case 2:
        Scanner.nTemplateType = 2003;
        break;
}
<extracts a template from captured fingerprint image>
ufs_res = Scanner.Extract(Template, out TemplateSize, out EnrollQuality);
if(ufs_res == UFS_STATUS.OK) {
    DrawCapturedImage(Scanner);
} else {
    UFScanner.GetErrorString(ufs_res, out m_strError);
    tbxMessage.AppendText("UFScanner Extract: " + m_strError + "\r\n");
    return;
}

<sets template type for verification>
switch (this.cbScanTemplateType.SelectedIndex)
{
    case 0:
        m_Matcher.nTemplateType = 2001;
        break;
    case 1:
        m_Matcher.nTemplateType = 2002;
        break;
    case 2:
        m_Matcher.nTemplateType = 2003;
        break;
}

<verifies first template is matched>
ufm_res = m_Matcher.Verify(Template, TemplateSize, m_template1[Serial],
m_template_size1[Serial], out VerifySucceed);
if(ufm_res != UFM_STATUS.OK) {
    UFMatcher.GetErrorString(ufm_res, out m_strError);
    tbxMessage.AppendText("UFMatcher Verify: " + m_strError + "\r\n");
    return;
}
if(VerifySucceed) {

```

```
    tbxMessage.AppendText("Verification succeed (No." + m_UserID[Serial] + ")\r\n");
} else {
    if(m_template_size2[Serial] != 0) {
        <verifies second template is matched>
        ufm_res = m_Matcher.Verify(Template, TemplateSize, m_template2[Serial],
m_template_size2[Serial], out VerifySucceed);
        if(ufm_res != UFM_STATUS.OK) {
            UFMatcher.GetErrorString(ufm_res, out m_strError);
            tbxMessage.AppendText("UFMatcher Verify: " + m_strError + "\r\n");
            return;
        }
        if(VerifySucceed) {
            tbxMessage.AppendText("Verification succeed (No." + m_UserID[Serial] +
")\r\n");
        }
        else {
            tbxMessage.AppendText("Verification failed\r\n");
        }
    }
    else {
        tbxMessage.AppendText("Verification failed\r\n");
    }
}
}
```

Identification

Description

A fingerprint can be identified compared with all saved fingerprints.

Demo sample's 'Identify' button is set to calling the OnIdentify(VS60) function. This function captures a fingerprint image and extracts a template from the image. And then calls the UFM_Identify function to execute 1:N matching using extracted template and all the other saved(enrolled) templates.

Using functions

[UFS_SetTemplateType](#), [UFM_SetTemplateType](#), [UFS_ClearCaptureImageBuffer](#),
[UFS_GetErrorString](#), [UFS_Extract](#), [UFM_Identify](#), [UFM_GetErrorString](#),
[UFS_GetScannerHandle](#) in GetCurrentScannerHandle

Source Code

VS60

```
void CUFE30_DemoDlg::OnIdentify()
{
    UpdateData(TRUE);
    HUFSscanner hScanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <ufm_res takes UFMatcher function's return value>
    UFM_STATUS ufm_res;
    unsigned char Template[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    int nEnrollQuality;
    char szType[20];
    char szMatchingType[20];
    int nMaxTemplateNum = 0;
    <gets a scanner handle>
    if (!GetCurrentScannerHandle(&hScanner)) {
        return;
    }

    unsigned char* template_all[MAX_TEMPLATE_NUM * 2];
    int templateSize_all[MAX_TEMPLATE_NUM * 2];
    int nIndex[MAX_TEMPLATE_NUM * 2];
    <initialize the template buffer>
    for (i = 0; i < m_template_num * 2; i++) {
        template_all[i] = (unsigned char*)malloc(MAX_TEMPLATE_SIZE);
        memset(template_all[i], 0, MAX_TEMPLATE_SIZE);
    }
}
```

```

        templateSize_all[i] = 0;
    }
    <merge the first template array and the second template array>
    for( i = 0; i < m_template_num * 2; i++) {
        if(i < m_template_num) {
            if(m_templateSize1[i] != 0) {
                memcpy(template_all[j], m_template1[i], m_templateSize1[i]);
                templateSize_all[j] = m_templateSize1[i];
                nIndex[j] = i;
                j++;
            }
        } else {
            if(m_templateSize2[i-m_template_num] != 0) {
                memcpy(template_all[j], m_template2[i-m_template_num],
m_templateSize2[i-m_template_num]);
                templateSize_all[j] = m_templateSize2[i-m_template_num];
                nIndex[j] = i - m_template_num;
                j++;
            }
        }
    }
    nMaxTemplateNum = j;

    <sets template type for extraction>
    switch (m_nType) {
        case 0:
            UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_SUPREMA);
            AddMessage("Current scanner,UFS_SetTemplateType: SUPREMA TYPE
\r\n");
            sprintf(szType,"%s","suprema type");
            break;
        case 1:
            UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_ISO19794_2);
            AddMessage("Current scanner,UFS_SetTemplateType: ISO_19794_2 TYPE
\r\n");
            sprintf(szType,"%s","iso_19794_2 type");
            break;
        case 2:
            UFS_SetTemplateType(hScanner, UFS_TEMPLATE_TYPE_ANSI378);
            AddMessage("Current scanner,UFS_SetTemplateType: ANSI378 TYPE \r\n");
            sprintf(szType,"%s","ansi378 type");
            break;
    }
    <sets template type for identification>
    switch (m_nMatchingType) {
        case 0:
            UFM_SetTemplateType(m_hMatcher,
UFM_TEMPLATE_TYPE_SUPREMA);
            AddMessage("Matching type,UFM_SetTemplateType: SUPREMA TYPE

```

```

\r\n");
sprintf(szMatchingType,"%s","suprema type");
break;
case 1:
    UFM_SetTemplateType(m_hMatcher,
    UFM_TEMPLATE_TYPE_ISO19794_2);
    AddMessage("Matching type,UFM_SetTemplateType: ISO_19794_2 TYPE
\r\n");
    sprintf(szMatchingType,"%s","iso_19794_2 type");
    break;
case 2:
    UFM_SetTemplateType(m_hMatcher, UFM_TEMPLATE_TYPE_ANSI378);
    AddMessage("Matching type,UFM_SetTemplateType: ANSI378 TYPE \r\n");
    sprintf(szMatchingType,"%s","ansi378 type");
    break;
}
UFS_ClearCaptureImageBuffer(hScanner);
AddMessage("Place a finger\r\n");
<captures a fingerprint image>
ufs_res = UFS_CaptureSingleImage(hScanner);
if(ufs_res != UFS_OK) {
<gets an error message string according to ufs_res which is a return value of UFS
function>
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_CaptureSingleImage: %s\r\n", m_strError);
    return;
}
<extracts a template from captured fingerprint image>
ufs_res = UFS_Extract(hScanner, Template, &TemplateSize, &nEnrollQuality);
if(ufs_res == UFS_OK) {
    Invalidate(FALSE);
} else {
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_Extract: %s\r\n", m_strError);
    return;
}
{
    int nMatchIndex;
    BeginWaitCursor();
    <identifies a fingerprint template against all saved fingerprint templates>
    ufm_res = UFM_Identify(m_hMatcher, Template, TemplateSize, template_all,
    templateSize_all, nMaxTemplateNum, 5000, &nMatchIndex);
    <identifies a fingerprint template against all saved fingerprint templates using
    multi-thread>
    //ufm_res = UFM_IdentifyMT(m_hMatcher, Template, TemplateSize,
    m_template, m_template_size, m_template_num, 5000, &nMatchIndex);
    EndWaitCursor();
    if(ufm_res != UFM_OK) {

```

```

    <gets the error message string according to ufm_res which is a return
     value of UFM function>
    UFM_GetErrorString(ufm_res, m_strError);
    AddMessage("UFM_Identify: %s\r\n", m_strError);
    return;
}
if(nMatchIndex != -1) {
    AddMessage("Identification succeed (Match Index.%d) (ID.%s)\r\n",
               nMatchIndex, m_userid[nindex[nMatchIndex]]);
} else {
    AddMessage("Identification failed\r\n");
}
}
}

```

VB60

```

Private Sub btnIdentify_Click()
    Dim hScanner As Long
        <ufs_res takes UFSanner function's return value>
    Dim ufs_res As UFS_STATUS
        <ufm_res takes UFMatcher function's return value>
    Dim ufm_res As UFM_STATUS
    Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
    Dim TemplateSize As Long
    Dim EnrollQuality As Long
    Dim MatchIndex As Long
    <gets a scanner handle>
    If (GetCurrentScannerHandle(hScanner) <> True) Then
        Exit Sub
    End If

    Dim template_all(MAX_TEMPLATE_SIZE - 1, (MAX_TEMPLATE_NUM * 2) - 1)
    As Byte
    Dim templateSize_all((MAX_TEMPLATE_NUM * 2) - 1) As Long
    Dim nIndex((MAX_TEMPLATE_NUM * 2) - 1) As Long
    Dim j As Long
    Dim i As Long
    Dim nMaxTemplateNum As Long

    j = 0
    nMaxTemplateNum = 0
    <merge the first template array and the second template array>
    For i = 0 To (m_template_num * 2) - 1
        If (i < m_template_num) Then
            If (m_templateSize1(i) <> 0) Then
                Call CopyMemory(Val VarPtr(template_all(0, j)), Val
                VarPtr(m_template1(0, i)), m_templateSize1(i))

```

```

templateSize_all(j) = m_templateSize1(i)
nindex(j) = i
j = j + 1
End If
Else
If (m_templateSize2(i - m_template_num) <> 0) Then
Call CopyMemory(ByVal VarPtr(template_all(0, j)), ByVal
VarPtr(m_template2(0, i - m_template_num)), m_templateSize2(i - m_template_num))
templateSize_all(j) = m_templateSize2(i - m_template_num)
nindex(j) = i - m_template_num
j = j + 1
End If
End If
Next
nMaxTemplateNum = j

<sets template type for extraction>
If (cbType.ListIndex = 0) Then
UFS_SetTemplateType hScanner,
UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_SUPREMA
ElseIf (cbType.ListIndex = 1) Then
UFS_SetTemplateType hScanner,
UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ISO19794_2
ElseIf (cbType.ListIndex = 2) Then
UFS_SetTemplateType hScanner,
UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ANSI378
End If

UFS_ClearCaptureImageBuffer (hScanner)
AddMessage ("Place your finger" & vbCrLf)

<captures a fingerprint image>
ufs_res = UFS_CaptureSingleImage(hScanner)
If (ufs_res <> UFS_STATUS.OK) Then
<gets the error message string according to ufs_res which is a return value of UFS
function>
UFS_GetErrorString ufs_res, m_strError
AddMessage ("UFS_CaptureSingleImage: " & m_strError & vbCrLf)
Exit Sub
End If

<extracts a template from captured fingerprint image>
ufs_res = UFS_Extract(hScanner, Template(0), TemplateSize, EnrollQuality)
If ufs_res = UFS_STATUS.OK Then
pbImageFrame.Refresh
Else
UFS_GetErrorString ufs_res, m_strError
AddMessage ("UFS_Extract: " & m_strError & vbCrLf)
Exit Sub

```

```

End If

' Make template pointer array to pass two dimensional template data
Dim template_ptr(MAX_TEMPLATE_NUM - 1) As Long
Dim i As Long
For i = 0 To MAX_TEMPLATE_NUM - 1
    template_ptr(i) = VarPtr(m_template(0, i))
Next

<sets template type for identification>
If (cbMatchType.ListIndex = 0) Then
    UFM_SetTemplateType m_hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_SUPREMA
ElseIf (cbType.ListIndex = 1) Then
    UFM_SetTemplateType m_hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_ISO19794_2
ElseIf (cbType.ListIndex = 2) Then
    UFM_SetTemplateType m_hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_ANSI378
End If

<identifies a fingerprint template against all saved fingerprint templates>
ufm_res = UFM_Identify(m_hMatcher, Template(0), TemplateSize, template_ptr(0),
templateSize_all(0), nMaxTemplateNum, 5000, MatchIndex)
If ufm_res <> UFM_STATUS.OK Then
    <gets the error message string according to ufm_res which is a return value of UFM
function>
    UFM_GetErrorString ufm_res, m_strError
    AddMessage ("UFM_Identify: " & m_strError & vbCrLf)
    Exit Sub
End If

If (MatchIndex <> -1) Then
    AddMessage ("Identification succeed (Match Index." & MatchIndex & ") (ID." &
m_UserID(nindex(MatchIndex)) & ")" & vbCrLf)
Else
    AddMessage ("Identification failed" & vbCrLf)
End If
End Sub

```

C#

```

private void btnIdentify_Click(object sender, EventArgs e) {
    UFSscanner Scanner;
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    <ufm_res takes UFMatcher function's return value>
    UFM_STATUS ufm_res;
    byte[] Template = new byte[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    int EnrollQuality;
    int MatchIndex;
    <gets a scanner handle>
    if (!GetGetCurrentScanner(out Scanner)) {
        return;
    }
    byte[][] template_all;
    int[] templateSize_all;
    int[] nindex;
    int i, j = 0, nMaxTemplateNum = 0;
    template_all = new byte[MAX_TEMPLATE_NUM * 2][];
    templateSize_all = new int[MAX_TEMPLATE_NUM * 2];
    nindex = new int[MAX_TEMPLATE_NUM * 2];
    <initialize the template buffer>
    for (i = 0; i < m_template_num * 2; i++) {
        template_all[i] = new byte[MAX_TEMPLATE_SIZE];
        templateSize_all[i] = 0;
    }
    <merge the first template array and the second template array>
    for (i = 0; i < m_template_num * 2; i++) {
        if (i < m_template_num) {
            if (m_template_size1[i] != 0) {
                System.Array.Copy(m_template1[i], 0, template_all[j], 0,
m_template_size1[i]);
                templateSize_all[j] = m_template_size1[i];
                nindex[j] = i;
                j++;
            }
        } else {
            if (m_template_size2[i - m_template_num] != 0) {
                System.Array.Copy(m_template2[i - m_template_num], 0, template_all[j],
0, m_template_size2[i - m_template_num]);
                templateSize_all[j] = m_template_size2[i - m_template_num];
                nindex[j] = i - m_template_num;
                j++;
            }
        }
    }
    nMaxTemplateNum = j;

    Scanner.ClearCaptureImageBuffer();
}

```

```

tbxMessage.AppendText("Place Finger\r\n");
<captures a fingerprint image>
ufs_res = Scanner.CaptureSingleImage();
if(ufs_res != UFS_STATUS.OK) {
    <gets an error message string according to ufs_res which is a return value of
    UFS function>
    UFScanner.GetErrorString(ufs_res, out m_strError);
    tbxMessage.AppendText("UFScanner CaptureSingleImage: " + m_strError +
    "\r\n");
    return;
}

switch (this.cbScanTemplateType.SelectedIndex)
{
    case 0:
        Scanner.nTemplateType = 2001;
        break;
    case 1:
        Scanner.nTemplateType = 2002;
        break;
    case 2:
        Scanner.nTemplateType = 2003;
        break;
}
<extracts a template from captured fingerprint image>
ufs_res = Scanner.Extract(Template, out TemplateSize, out EnrollQuality);
if(ufs_res == UFS_STATUS.OK) {
    DrawCapturedImage(Scanner);
} else {
    UFScanner.GetErrorString(ufs_res, out m_strError);
    tbxMessage.AppendText("UFScanner Extract: " + m_strError + "\r\n");
    return;
}
<sets template type for identification>
switch (this.cbScanTemplateType.SelectedIndex)
{
    case 0:
        m_Matcher.nTemplateType = 2001;
        break;
    case 1:
        m_Matcher.nTemplateType = 2002;
        break;
    case 2:
        m_Matcher.nTemplateType = 2003;
        break;
}
Cursor.Current = Cursors.WaitCursor;
/*
<identifies a fingerprint template against all saved fingerprint templates>

```

```

ufm_res = m_Matcher.Identify(Template, TemplateSize, template_all,
templateSize_all, nMaxTemplateNum, 5000, out MatchIndex);
//ufm_res = m_Matcher.IdentifyMT(Template, TemplateSize, m_template,
templateSize_all, nMaxTemplateNum, 5000, out MatchIndex);
/*
{
byte[,] Template2 = new byte[m_template_num, MAX_TEMPLATE_SIZE];
int i, j;
for (j = 0; j < m_template_num; j++) {
    for (i = 0; i < m_template_size[j]; i++) {
        Template2[j,i] = m_template[j][i];
    }
}
ufm_res = m_Matcher.Identify(Template, TemplateSize, m_template,
m_template_size, m_template_num, 5000, out MatchIndex);
}
*/
Cursor.Current = this.Cursor;
if(ufm_res != UFM_STATUS.OK) {
    <gets an error message string according to ufm_res which is a return value of
    UFM function>
    UFMatcher.GetErrorString(ufm_res, out m_strError);
    tbxMessage.AppendText("UFMatcher Identify: " + m_strError + "\r\n");
    return;
}
if(MatchIndex != -1) {
    tbxMessage.AppendText("Identification succeed (Match Index." + MatchIndex
    + ") (ID." + m UserID[nindex[MatchIndex]] + ")\r\n");
} else {
    tbxMessage.AppendText("Identification failed\r\n");
}
}

```

UFE30_ImageDemo

UFE30_ImageDemo provides the demo about extracting templates from images and matching two templates. This application uses [UFMatcher](#) and [UFExtractor](#) modules.

Required products

Products	Remarks
Suprema Image SDK	No restriction

Available languages

Languages	Locations
Visual C++ 6.0	samples\VS60\UFE30_ImageDemoVC60
Visual Basic 6.0	samples\VS60\UFE30_ImageDemoVB60
Visual C#	samples\VS80\UFE30_ImageDemoCS
Visual Basic .NET	samples\VS80\UFE30_ImageDemoVBNET

UFE30_ImageDemo_Usage

Executable File Location

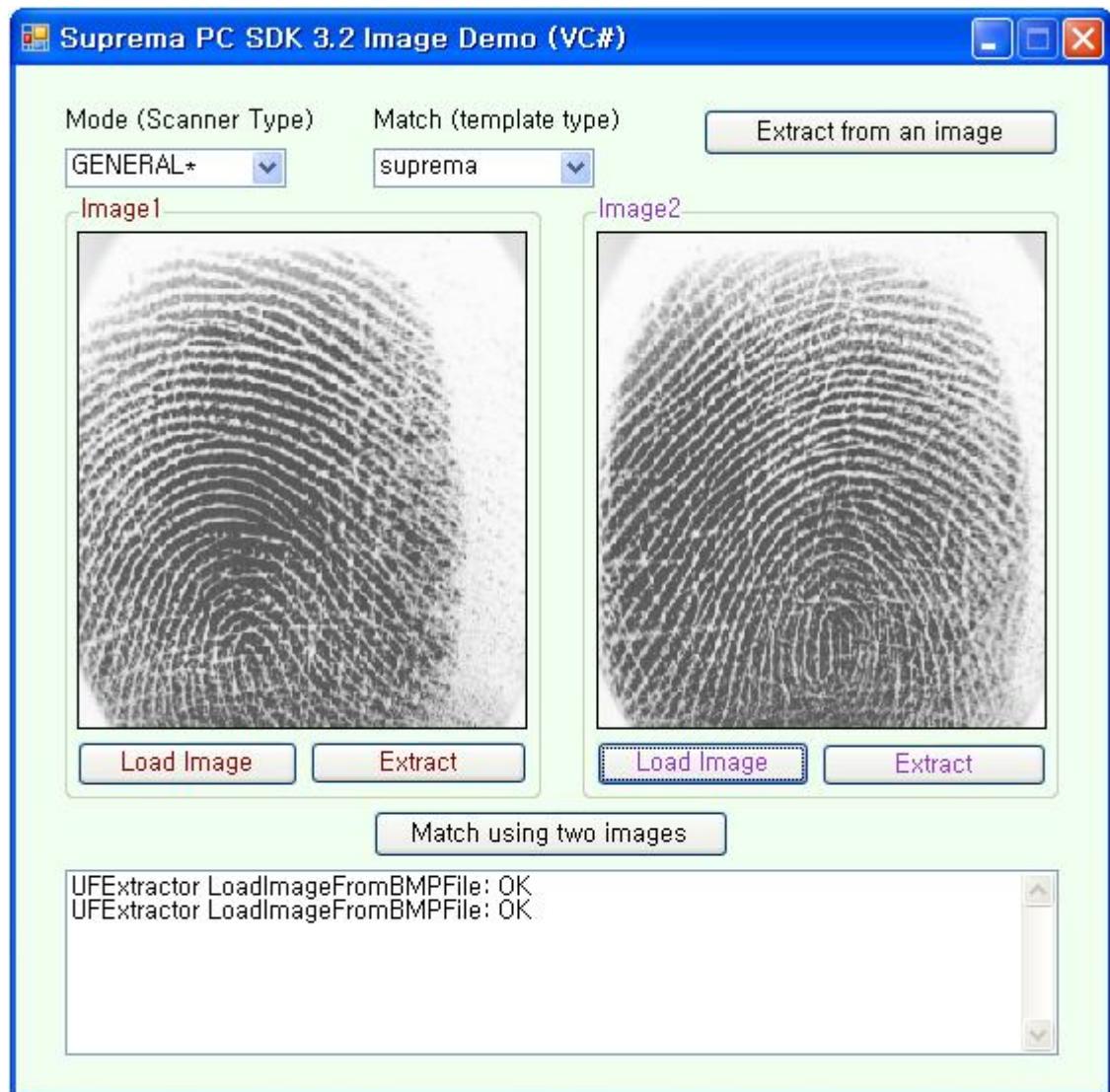
- bin\UFE30_ImageDemoCS60.exe
- bin\UFE30_ImageDemoVBNET.exe
- bin\UFE30_ImageDemoCS.exe
- Source code of all demo application

Required products

[Suprema Image SDK](#)

Picture of the ImageDemo sample applications







User Interface Component

Suprema PC SDK ImageDemo sample demonstrates how to use Suprema Image SDK's functions using bitmap image file as resource. Please use this sample as a reference for making your own program. Image SDK demo provides following methods:

- Load bitmap image - A fingerprint image file can be loaded to display at the image frame component and extract a template.
- Extract template - Extracts a template which is the feature data interchange format from a fingerprint image file
- Matching - Confirms two fingerprint images whether matched or not.

Every UFE30_ImageDemo application is written by different languages from Visual C++ to C#. But they are made of same functions and same interface design.

The picture bellow shows the PC SDK ImageDemo sample main window and description about bunch of interface components :



The table bellow shows the Image SDK ImageDemo's simple functionality and links to each detailed usage page :

Image
Loads an image file using the Load Image button and extracts a template from the loaded image using 'Extract' button
Load Image button - Loads a fingerprint image which is the bmp format or the wsq format file Source Code
Extract button - Extracts a template from loaded image Source Code

Match
Checks two fingerprint images whether matched or not
Match using two images button - Verifies whether two loaded fingerprint images are matched Source Code

Template setting
Sets the scanner type and the template type for extracting process
Scanner Type combo box - Sets the type of scanner that captured the loaded image. Source Code
Template Type combo box - Sets the template type Source Code

Extract single

Extracts a template from the fingerprint image
Extract from an image button - Extracts a template from an image which is selected in file dialog <u>Source Code</u>

Load Image File

Description

Image sample's 'Load Image' button is set to calling the OnImageButton(VS60) function. It declares a variable for image's location and then calls the UFE_LoadImageFromBMPFile function to load a fingerprint image. See also the Onpaint(VS60) function, which is event function, is called when the dialog drawing event occurred. This function displays the fingerprint image at the image frame component by using UFE_DrawImage function.

Using functions

[UFE_LoadImageFromBMPFile](#), [UFE_GetErrorString](#)

Source Code

VS60

```
void CUFE30_ImageDemoDlg::OnImageButton()
{
    // TODO: Add your control notification handler code here
    <ufe_res takes UFExtrator function's return value>
    UFE_STATUS ufe_res;

    <makes a filedialog to get an image file location>
    CFileDialog dlg(TRUE, "bmp", NULL, NULL, "Bitmap Files (*.bmp)|*.bmp");
    if(dlg.DoModal() != IDOK) {
        AddMessage("Image file selection is cancelled by user\r\n");
        return;
    }

    // Load image
    <Loads an image data>
    ufe_res = UFE_LoadImageFromBMPFile(dlg.GetPathName(), m_pImage_1,
    &m_nWidth_1, &m_nHeight_1);
    if(ufe_res == UFE_OK) {
        AddMessage("UFE_LoadImageFromBMPFile from(%s): OK\r\n",
        dlg.GetFileName());
        m_fileon_1 = true;
        this->m_szFilename_1 = dlg.GetPathName();
    } else {
        <gets an error message string according to ufe_res which is a return value of
        UFE function>
```

```

        UFE_GetErrorString(ufe_res, m_strError);
        AddMessage("UFE_LoadImageFromBMPFile: %s\r\n", m_strError);
    }
    Invalidate(false);
}

```

VB60

```

Private Sub bt_Load1_Click()

    cdFileDialog.Filter = "Bitmap Files (*.bmp)|*.bmp"
    cdFileDialog.DefaultExt = "bmp"
    cdFileDialog.Flags = cdlOFNHideReadOnly Or cdlOFNPathMustExist Or
    cdlOFNOverwritePrompt Or cdlOFNNoReadOnlyReturn
    cdFileDialog.ShowOpen
        <makes a filedialog to get an image file location>
    If (cdFileDialog.FileName = "") Then
        AddMessage ("Image file selection is cancelled by user" & vbCrLf)
        Exit Sub
    End If
    m_szFilename1 = cdFileDialog.FileName

    '=====
    ' Load image
    '=====

    <ufe_res takes UFExtracter function's return value>
    Dim ufe_res As Long

    <saves an image data to m_pImage_1>
    ufe_res = UFE_LoadImageFromBMPFile(cdFileDialog.FileName, m_pImage_1(0),
    m_nWidth_1, m_nHeight_1)
    If (ufe_res = UFE_STATUS.OK) Then
        AddMessage ("UFE_LoadImageFromBMPFile: OK" & vbCrLf)
        Picture1.Refresh
    Else
        <gets an error message string according to ufe_res which is a return value of
        UFE function>
        UFE_GetErrorString ufe_res, m_strError
        AddMessage ("UFE_LoadImageFromBMPFile: " & m_strError & vbCrLf)
        'ExtractTemplate = False
        Exit Sub
    End If
    '=====
End Sub

```

C#

```

private void btnLoad1_Click(object sender, EventArgs e)
{
    <ufe_res takes UFExtractor function's return value>
    UFE_STATUS ufe_res;
    byte[] ImageData = null;
    int Width;
    int Height;
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "Bitmap files (*.bmp)|*.bmp";
    dlg.DefaultExt = "bmp";
    <makes a filedialog to get an image file location>
    if(dlg.ShowDialog() != DialogResult.OK)
    {
        tbxMessage.AppendText("Image file selection is cancelled by user\r\n");
        return;
    }
    m_szFilename1 = dlg.FileName;
    // Load image
    // Load image
    <saves an image data to the ImageData variable>
    ufe_res = UFExtractor.LoadImageFromBMPFile(dlg.FileName, out ImageData, out
    Width, out Height);
    if(ufe_res == UFE_STATUS.OK)
    {
        tbxMessage.AppendText("UFExtractor LoadImageFromBMPFile: OK\r\n");
    }
    else
    {
        <gets an error message string according to ufe_res which is a return value of
        UFE function>
        UFExtractor.GetErrorString(ufe_res, out m_strError);
        tbxMessage.AppendText("UFExtractor LoadImageFromBMPFile: " +
        m_strError + "\r\n");
        return;
    }
    // Draw image
    Graphics g = pictureBox1.CreateGraphics();
    Rectangle rect = new Rectangle(0, 0, pictureBox1.Width, pictureBox1.Height);
    try
    {
        <draws the loaded image at pictureBox1>
        UFExtractor.DrawImage(g, rect, ImageData, Width, Height);
    }
    finally
    {

```

```
        g.Dispose();
    }
///////////////////////////////////////////////////////////////////
```

Scanner Setting

Description

This function can be used before extraction to set the optimal environment. Set this option according to the scanner type, which captured the fingerprint image. To get the best quality template from UFE_Extract function. If you get the image from our scanner, set this value depending on your scanner type. And if you get the fingerprint image from other company's product, set this value 'GENERAL'.

Using functions

[UFE_SetMode](#)

Source Code

VS60

```
switch (m_nMode) {
    case 0:
        UFE_SetMode(hExtractor, UFE_MODE_SFR200);
        break;
    case 1:
        UFE_SetMode(hExtractor, UFE_MODE_SFR300);
        break;
    case 2:
        UFE_SetMode(hExtractor, UFE_MODE_SFR300v2);
        break;
    case 3:
        UFE_SetMode(hExtractor, UFE_MODE_GENERAL);
        break;
}
```

VB60

```
If (cbMode.ListIndex = 0) Then
    UFE_SetMode hExtractor, UFE_MODE.SFR200
ElseIf (cbMode.ListIndex = 1) Then
    UFE_SetMode hExtractor, UFE_MODE.SFR300
ElseIf (cbMode.ListIndex = 2) Then
    UFE_SetMode hExtractor, UFE_MODE.SFR300v2
ElseIf (cbMode.ListIndex = 3) Then
    UFE_SetMode hExtractor, UFE_MODE.General
End If
```

C#

```
switch (cbMode.SelectedIndex) {
    case 0:
        Extractor.Mode = UFE_MODE.SFR200;
        break;
    case 1:
        Extractor.Mode = UFE_MODE.SFR300;
        break;
    case 2:
        Extractor.Mode = UFE_MODE.SFR300v2;
        break;
    case 3:
        Extractor.Mode = UFE_MODE.GENERAL;
        break;
}
```

Template Type Setting

Description

The UFE_SetTemplateType can be used before the extraction to set the type of template. Use this function to set the template type that you want to get from your image file. The default type is 'SUPREMA' type.

Using functions

UFE_SetTemplateType

Source Code

VS60

```
switch (m_nType) {
    case 0:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_SUPREMA);
        break;
    case 1:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_ISO19794_2);
        break;
    case 2:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_ANSI378);
        break;
}
```

VB60

```
If (cbType.ListIndex = 0) Then
    UFM_SetTemplateType hMatcher,
    UFM_TEMPLATE_TYPE.UMF_TEMPLATE_TYPE_SUPREMA
ElseIf (cbType.ListIndex = 1) Then
    UFM_SetTemplateType hMatcher,
    UFM_TEMPLATE_TYPE.UMF_TEMPLATE_TYPE_ISO19794_2
ElseIf (cbType.ListIndex = 2) Then
    UFM_SetTemplateType hMatcher,
    UFM_TEMPLATE_TYPE.UMF_TEMPLATE_TYPE_ANSI378
End If
```

C#

```
switch (cbTemplateType.SelectedIndex){
    case 0:
        Extractor.nTemplateType = 2001;
        break;
    case 1:
```

```
Extractor.nTemplateType = 2002;  
break;  
case 2:  
Extractor.nTemplateType = 2003;  
break;  
}
```

Extract a template from displayed image

Description

Image demo sample's 'Extract' button is set to calling the OnExtractButton(VS60) function. This function extracts a template from an image file, which was loaded at the image frame by using UFE_Extract function.

Using functions

[UFE_GetErrorString](#), [UFE_Create](#), [UFE_SetParameter](#), [UFE_SetMode](#),
[UFE_SetTemplateType](#), [UFE_Delete](#)

Source Code

VS60

```
void CUFE30_ImageDemoDlg::OnExtractButton1()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE);
    HUFExtractor hExtractor;
    <ufe_res takes UFExtractor function's return value>
    UFE_STATUS ufe_res;
    unsigned char Template[MAX_TEMPLATE_SIZE];
    int nTemplateSize;
    int nValue;
    /////////////////////////////////
    // Create extractor
    ///////////////////////////////
    <creates an extractor>
    ufe_res = UFE_Create(&hExtractor);
    if(ufe_res == UFE_OK) {
        AddMessage("UFE_Create: OK\r\n");
    } else {
        <gets an error message string according to ufe_res which is a return value of
        UFE function>
        UFE_GetErrorString(ufe_res, m_strError);
        AddMessage("UFE_Create: %s\r\n", m_strError);
        return;
    }
    ///////////////////////////////
    // Set Parameters and Mode for extractor
    ///////////////////////////////
    <extraction settings. see this page to check about 'detect core'>
    nValue = 0;
```

```

UFE_SetParameter(hExtractor, UFE_PARAM_DETECT_CORE, &nValue);
nValue = MAX_TEMPLATE_SIZE;
UFE_SetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE, &nValue);
<sets the scanner type>
switch (m_nMode) {
    case 0:
        UFE_SetMode(hExtractor, UFE_MODE_SFR200);
        break;
    case 1:
        UFE_SetMode(hExtractor, UFE_MODE_SFR300);
        break;
    case 2:
        UFE_SetMode(hExtractor, UFE_MODE_SFR300v2);
        break;
    case 3:
        UFE_SetMode(hExtractor, UFE_MODE_GENERAL);
        break;
}
<sets the template type>
switch (m_nType) {
    case 0:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_SUPREMA);
        break;
    case 1:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_ISO19794_2);
        break;
    case 2:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_ANSI378);
        break;
}
///////////
<extracts a template. the m_szFilename_1 is location of image. The extracted
template data will be saved in the 'Template' variable>
if (ExtractTemplate(hExtractor, Template, &nTemplateSize, m_szFilename_1)) {
    AddMessage("Template extraction is succeeded\r\n");
}
///////////
// Delete extractor
///////////
if (hExtractor != NULL) {
    ufe_res = UFE_Delete(hExtractor);
    if (ufe_res == UFE_OK) {
        AddMessage("UFE_Delete: OK\r\n");
    } else {
        UFE_GetErrorString(ufe_res, m_strError);
        AddMessage("UFE_Delete: %s\r\n", m_strError);
    }
}
}

```

VB60

```

Private Sub bt_ext1_Click()
    Dim hExtractor As Long
        <ufe_res takes UFExtractor function's return value>
    Dim ufe_res As UFE_STATUS
    Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
    Dim TemplateSize As Long

    '=====
    ' Create extractor
    '=====

    <creates an extractor>
    ufe_res = UFE_Create(hExtractor)
    If (ufe_res = UFE_STATUS.OK) Then
        AddMessage ("UFE_Create: OK" & vbCrLf)
    Else
        <gets an error message string according to ufe_res which is a return value of UFE
        function>
        UFE_GetErrorString ufe_res, m_strError
        AddMessage ("UFE_Create: " & m_strError & vbCrLf)
    End If
    '=====

    '=====
    ' Set Parameters and Mode for extractor
    '=====

    <extraction settings. see this page to check about 'detect core'>
    UFE_SetParameter hExtractor, UFE_PARAM.DETECT_CORE, 0
    UFE_SetParameter hExtractor, UFE_PARAM.TEMPLATE_SIZE,
MAX_TEMPLATE_SIZE
    UFE_SetParameter hExtractor, UFE_PARAM.USE_SIF, 0

    <sets the scanner type>
    If (cbMode.ListIndex = 0) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR200
    ElseIf (cbMode.ListIndex = 1) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR300
    ElseIf (cbMode.ListIndex = 2) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR300v2
    ElseIf (cbMode.ListIndex = 3) Then
        UFE_SetMode hExtractor, UFE_MODE.General
    End If

```

```

<sets the template type>
If(cbType.ListIndex = 0) Then
    UFE_SetTemplateType hExtractor,
    UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_SUPREMA
ElseIf(cbType.ListIndex = 1) Then
    UFE_SetTemplateType hExtractor,
    UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_ISO19794_2
ElseIf(cbType.ListIndex = 2) Then
    UFE_SetTemplateType hExtractor,
    UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_ANSI378
End If
'=====
====='

If(m_szFilename1 = "") Then
    AddMessage ("Image file selection is cancelled by user" & vbCrLf)
    GoTo errret
End If

<extracts a template. m_szFilename_1 is location of image. The extracted template data
will be saved in the 'Template' variable>
If(ExtractTemplate(hExtractor, Template, TemplateSize, m_szFilename1)) Then
    AddMessage ("Template extraction is succeeded" & vbCrLf)
End If
errret:
'=====
====='

'Delete extractor
'=====
====='

ufe_res = UFE_Delete(hExtractor)
If(ufe_res = UFE_STATUS.OK) Then
    AddMessage ("UFE_Delete: OK" & vbCrLf)
Else
    UFE_GetErrorString ufe_res, m_strError
    AddMessage ("UFE_Delete: " & m_strError & vbCrLf)
End If
'=====
====='

End Sub

```

```
C#
private void btnExtact_Click(object sender, EventArgs e)
{
    UFExtractor Extractor = null;
    byte[] Template = new byte[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    /////////////////////////////////
    // Create extractor
    ///////////////////////////////
    <creates an extractor>
    Extractor = new UFExtractor();
    /////////////////////////////////
    ///////////////////////////////
    // Set Parameters and Mode for extractor
    /////////////////////////////////
    <extraction settings. see this page to check about 'detect core'>
    Extractor.DetectCore = false;
    Extractor.TemplateSize = MAX_TEMPLATE_SIZE;
    Extractor.UseSIF = false;
    <sets the scanner type>
    switch (cbMode.SelectedIndex) {
        case 0:
            Extractor.Mode = UFE_MODE.SFR200;
            break;
        case 1:
            Extractor.Mode = UFE_MODE.SFR300;
            break;
        case 2:
            Extractor.Mode = UFE_MODE.SFR300v2;
            break;
        case 3:
            Extractor.Mode = UFE_MODE.GENERAL;
            break;
    }
    ///////////////////////////////
    <extracts a template. m_szFilename1 is location of image. The extracted template
    data will be saved in the 'Template' variable>
    if(ExtractTemplate(ref Extractor, Template, out TemplateSize, m_szFilename1))
    {
        tbxMessage.AppendText("Template extraction is succeeded\r\n");
    }
}
```

Extract a template from selecting image

Description

Image demo sample's 'Extract from an image' button is set to calling the OnExtract(VS60) function. This function extracts a template from an image file by using the UFE_Extract function. See the source code below, OnExtract function extracts a template from selected fingerprint image at the filedialog.

Using functions

[UFE_GetErrorString](#), [UFE_Create](#), [UFE_SetParameter](#), [UFE_SetMode](#),
[UFE_SetTemplateType](#), [UFE_Delete](#)

Source Code

VS60

```
void CUFE30_ImageDemoDlg::OnExtract()
{
    UpdateData(TRUE);
    HUFExtractor hExtractor;
    <ufe_res takes UFExtractor function's return value>
    UFE_STATUS ufe_res;
    unsigned char Template[MAX_TEMPLATE_SIZE];
    int nTemplateSize;
    int nValue;
    // Create extractor
    <creates an extractor>
    ufe_res = UFE_Create(&hExtractor);
    if(ufe_res == UFE_OK) {
        AddMessage("UFE_Create: OK\r\n");
    } else {
        <gets an error message string according to ufe_res which is a return value of
        UFE function>
        UFE_GetErrorString(ufe_res, m_strError);
        AddMessage("UFE_Create: %s\r\n", m_strError);
        return;
    }
    // Set Parameters and Mode for extractor
    <extraction settings. see this page to check about 'detect core'>
```

```

nValue = 0;
UFE_SetParameter(hExtractor, UFE_PARAM_DETECT_CORE, &nValue);
nValue = MAX_TEMPLATE_SIZE;
UFE_SetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE, &nValue);
<sets the scanner type>
switch (m_nMode) {
    case 0:
        UFE_SetMode(hExtractor, UFE_MODE_SFR200);
        break;
    case 1:
        UFE_SetMode(hExtractor, UFE_MODE_SFR300);
        break;
    case 2:
        UFE_SetMode(hExtractor, UFE_MODE_SFR300v2);
        break;
    case 3:
        UFE_SetMode(hExtractor, UFE_MODE_GENERAL);
        break;
}
<sets the template type>
switch (m_nType) {
    case 0:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_SUPREMA);
        break;
    case 1:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_ISO19794_2);
        break;
    case 2:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_ANSI378);
        break;
}
////////////////////////////////////////////////////////////////
// Load image
////////////////////////////////////////////////////////////////
<makes a file dialog to get an image file location>
CFileDialog dlg(TRUE, "bmp", NULL, NULL, "Bitmap Files (*.bmp)|*.bmp");
if(dlg.DoModal() != IDOK) {
    AddMessage("Image file selection is cancelled by user\r\n");
    return;
}
<extracts a template. The extracted template data will be saved in the 'Template' variable>
if(ExtractTemplate(hExtractor, Template, &nTemplateSize, dlg.GetPathName())) {
    AddMessage("Template extraction is succeeded\r\n");
}
////////////////////////////////////////////////////////////////
// Delete extractor

```

```
//////////  
ufe_res = UFE_Delete(hExtractor);  
if(ufe_res == UFE_OK) {  
    AddMessage("UFE_Delete: OK\r\n");  
} else {  
    UFE_GetErrorString(ufe_res, m_strError);  
    AddMessage("UFE_Delete: %s\r\n", m_strError);  
    return;  
}  
}
```

VB60

```

Private Sub Extract_Click()
    Dim hExtractor As Long
        <ufe_res takes UFExtractor function's return value>
    Dim ufe_res As UFE_STATUS
    Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
    Dim TemplateSize As Long

    '
    ====='
    ' Create extractor
    '
    ====='
        <creates an extractor>
    ufe_res = UFE_Create(hExtractor)
    If (ufe_res = UFE_STATUS.OK) Then
        AddMessage ("UFE_Create: OK" & vbCrLf)
    Else
        <gets an error message string according to ufe_res which is a return value of
        UFE function>
        UFE_GetErrorString ufe_res, m_strError
        AddMessage ("UFE_Create: " & m_strError & vbCrLf)
    End If
    '
    ====='

    '
    ====='
    ' Set Parameters and Mode for extractor
    '
    ====='
        <extraction settings. see this page to check about 'detect core'>
    UFE_SetParameter hExtractor, UFE_PARAM.DETECT_CORE, 0
    UFE_SetParameter hExtractor, UFE_PARAM.TEMPLATE_SIZE,
MAX_TEMPLATE_SIZE
    UFE_SetParameter hExtractor, UFE_PARAM.USE_SIF, 0

        <sets the scanner type>
    If (cbMode.ListIndex = 0) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR200
    ElseIf (cbMode.ListIndex = 1) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR300
    ElseIf (cbMode.ListIndex = 2) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR300v2
    ElseIf (cbMode.ListIndex = 3) Then
        UFE_SetMode hExtractor, UFE_MODE.General
    End If

        <sets the template type>
    If (cbType.ListIndex = 0) Then

```

```

        UFE_SetTemplateType hExtractor,
UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_SUPREMA
    ElseIf(cbType.ListIndex = 1) Then
        UFE_SetTemplateType hExtractor,
UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_ISO19794_2
    ElseIf(cbType.ListIndex = 2) Then
        UFE_SetTemplateType hExtractor,
UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_ANSI378
    End If
'=====
=====
'

'=====
=====

<makes a filedialog to get an image file location>
cdFileDialog.Filter = "Bitmap Files (*.bmp)|*.bmp"
cdFileDialog.DefaultExt = "bmp"
cdFileDialog.Flags = cdlOFNHideReadOnly Or cdlOFNPathMustExist Or
cdlOFNOverwritePrompt Or cdlOFNNoReadOnlyReturn
cdFileDialog.ShowOpen
If(cdFileDialog.FileName = "") Then
    AddMessage ("Image file selection is cancelled by user" & vbCrLf)
    GoTo errret
End If
m_szFilename1 = cdFileDialog.FileName
'=====

<extracts a template. The extracted template data will be saved in the 'Template'
variable>
If(ExtractTemplate(hExtractor, Template, TemplateSize, cdFileDialog.FileName))
Then
    AddMessage ("Template extraction is succeeded" & vbCrLf)
End If
errret:
'=====
=====
'

'Delete extractor
'=====

ufe_res = UFE_Delete(hExtractor)
If(ufe_res = UFE_STATUS.OK) Then
    AddMessage ("UFE_Delete: OK" & vbCrLf)
Else
    UFE_GetErrorString ufe_res, m_strError
    AddMessage ("UFE_Delete: " & m_strError & vbCrLf)
End If
'=====

End Sub

```

C#

```

private void btnExtract_Click(object sender, EventArgs e) {
    UFExtractor Extractor = null;
    byte[] Template = new byte[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    /////////////////////////////////
    // Create extractor
    ///////////////////////////////
    <creates an extractor>
    Extractor = new UFExtractor();
    ///////////////////////////////
    ///////////////////////////////
    // Set Parameters and Mode for extractor
    ///////////////////////////////
    <extraction settings. see this page to check about 'detect core'>
    Extractor.DetectCore = false;
    Extractor.TemplateSize = MAX_TEMPLATE_SIZE;
    Extractor.UseSIF = false;

    <sets the scanner type>
    switch (cbMode.SelectedIndex) {
        case 0:
            Extractor.Mode = UFE_MODE.SFR200;
            break;
        case 1:
            Extractor.Mode = UFE_MODE.SFR300;
            break;
        case 2:
            Extractor.Mode = UFE_MODE.SFR300v2;
            break;
        case 3:
            Extractor.Mode = UFE_MODE.GENERAL;
            break;
    }
    ///////////////////////////////
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "Bitmap files (*.bmp)|*.bmp";
    dlg.DefaultExt = "bmp";
    <makes a filedialog to get an image file location>
    if (dlg.ShowDialog() != DialogResult.OK)
    {
        tbxMessage.AppendText("Image file selection is cancelled by user\r\n");
        return;
    }
    <extracts a template. The extracted template data will be saved in the 'Template'
variable>
    if (ExtractTemplate(ref Extractor, Template, out TemplateSize, dlg.FileName)) {

```

```
        tbxMessage.AppendText("Template extraction is succeeded\r\n");  
    }  
}
```

Match Two Images

Description

Image demo sample's 'Match using two images' button is set to calling the OnMatch(VS60) function. It verifies Verifying two images whether they are matched or not by using UFE_Verify function. This action should be performed after two fingerprint images loaded.

Using functions

[UFE_GetErrorString](#), [UFE_Create](#), [UFE_SetParameter](#), [UFE_SetMode](#),
[UFE_SetTemplateType](#), [UFE_Delete](#), [UFM_Verify](#), [UFM_GetErrorString](#), [UFM_Create](#),
[UFM_SetParameter](#)

Source Code

VS60
<pre>void CUFE30_ImageDemoDlg::OnMatch() { UpdateData(TRUE); HUFExtractor hExtractor = NULL; HUFMatcher hMatcher = NULL; <ufe_res takes UFExtractor function's return value> UFE_STATUS ufe_res; <ufm_res takes UFMatcher function's return value> UFM_STATUS ufm_res; unsigned char Template1[MAX_TEMPLATE_SIZE]; unsigned char Template2[MAX_TEMPLATE_SIZE]; int nTemplate1Size; int nTemplate2Size; int bVerifySucceed; int nValue; CString log; // Create extractor <creates an extractor> ufe_res = UFE_Create(&hExtractor); if(ufe_res == UFE_OK) { AddMessage("UFE_Create: OK\r\n"); } else { <gets an error message string according to ufe_res which is a return value of UFE function> UFE_GetErrorString(ufe_res, m_strError); AddMessage("UFE_Create: %s\r\n", m_strError); goto errret; } }</pre>

```

}

///////////////////////////////
// Set Parameters and Mode for extractor
/////////////////////////////
<extraction settings. see this page to check 'detect core'>
nValue = 0;
UFE_SetParameter(hExtractor, UFE_PARAM_DETECT_CORE, &nValue);
nValue = MAX_TEMPLATE_SIZE;
UFE_SetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE, &nValue);

<sets the scanner type>
switch (m_nMode) {
    case 0:
        UFE_SetMode(hExtractor, UFE_MODE_SFR200);
        break;
    case 1:
        UFE_SetMode(hExtractor, UFE_MODE_SFR300);
        break;
    case 2:
        UFE_SetMode(hExtractor, UFE_MODE_SFR300v2);
        break;
    case 3:
        UFE_SetMode(hExtractor, UFE_MODE_GENERAL);
        break;
}
<sets the template type>
switch (m_nType) {
    case 0:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_SUPREMA);
        break;
    case 1:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_ISO19794_2);
        break;
    case 2:
        UFE_SetTemplateType(hExtractor, UFE_TEMPLATE_TYPE_ANSI378);
        break;
}
/////////////////////////////
// Create matcher
/////////////////////////////
<creates a matcher>
ufm_res = UFM_Create(&hMatcher);
if (ufm_res == UFM_OK) {
    AddMessage("UFM_Create: OK\r\n");
} else {

```

```

        UFM_GetErrorString(ufm_res, m_strError);
        AddMessage("UFM_Create: %s\r\n", m_strError);
        goto errret;
    }
    ///////////////////////////////////////////////////////////////////
    // Set Parameters for matcher
    ///////////////////////////////////////////////////////////////////
    nValue = 4;
    <matcher settings. see this page to check 'security level'>
    UFM_SetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL, &nValue);
    //nValue = 0;
    //UFM_SetParameter(hMatcher, UFM_PARAM_USE_SIF, &nValue);
    ///////////////////////////////////////////////////////////////////
    <template type setting. This must be same with extracted template>
    switch (m_nType) {
        case 0:
            UFM_SetTemplateType(hMatcher, UFM_TEMPLATE_TYPE_SUPREMA);
            break;
        case 1:
            UFM_SetTemplateType(hMatcher, UFM_TEMPLATE_TYPE_ISO19794_2);
            break;
        case 2:
            UFM_SetTemplateType(hMatcher, UFM_TEMPLATE_TYPE_ANSI378);
            break;
    }
    <extracts two templates from two loaded images>
    if (!ExtractTemplate(hExtractor, Template1, &nTemplate1Size, m_szFilename_1)
    || !ExtractTemplate(hExtractor, Template2, &nTemplate2Size, m_szFilename_2)) {
        goto errret;
    }
    <verifies two fingerprint images whether matched or not>
    ufm_res = UFM_Verify(hMatcher, Template1, nTemplate1Size, Template2,
    nTemplate2Size, &bVerifySucceed);
    if (ufm_res != UFM_OK) {
        UFM_GetErrorString(ufm_res, m_strError);
        AddMessage("UFM_Verify: %s\r\n", m_strError);
        goto errret;
    }
    if (bVerifySucceed) {
        AddMessage("Verification succeed\r\n");
    } else {
        AddMessage("Verification failed\r\n");
    }
errret:
    ///////////////////////////////////////////////////////////////////
    // Delete extractor

```

```
///////////
if(hExtractor != NULL) {
    ufe_res = UFE_Delete(hExtractor);
    if(ufe_res == UFE_OK) {
        AddMessage("UFE_Delete: OK\r\n");
    } else {
        UFE_GetErrorString(ufe_res, m_strError);
        AddMessage("UFE_Delete: %s\r\n", m_strError);
    }
}
///////////
// Delete matcher
///////////
if(hMatcher != NULL) {
    ufm_res = UFM_Delete(hMatcher);
    if(ufm_res == UFM_OK) {
        AddMessage("UFM_Delete: OK\r\n");
    } else {
        UFM_GetErrorString(ufm_res, m_strError);
        AddMessage("UFM_Delete: %s\r\n", m_strError);
    }
}
/////////
}
```

VB60

```

Private Sub Match_Click()
    Dim hExtractor As Long
    Dim hMatcher As Long
        <ufe_res takes UFExtractor function's return value>
    Dim ufe_res As UFE_STATUS
        <ufm_res takes UFMATCHER function's return value>
    Dim ufm_res As UFM_STATUS
    Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
    Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
    Dim Template1Size As Long
    Dim Template2Size As Long
    Dim VerifySucceed As Long

    hExtractor = 0
    hMatcher = 0
    '=====
    ====='
    ' Create extractor
    '=====
    ====='
        <creates an extractor>
    ufe_res = UFE_Create(hExtractor)
    If (ufe_res = UFE_STATUS.OK) Then
        AddMessage ("UFE_Create: OK" & vbCrLf)
    Else
        UFE_GetErrorString ufe_res, m_strError
        AddMessage ("UFE_Create: " & m_strError & vbCrLf)
    End If
    '=====
    ====='
    ' Set Parameters and Mode for extractor
    '=====
    ====='
        <extraction settings. see this page to check 'detect core'>
    UFE_SetParameter hExtractor, UFE_PARAM.DETECT_CORE, 0
    UFE_SetParameter hExtractor, UFE_PARAM.TEMPLATE_SIZE,
MAX_TEMPLATE_SIZE
    UFE_SetParameter hExtractor, UFE_PARAM.USE_SIF, 0
        <sets the scanner type>
    If (cbMode.ListIndex = 0) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR200
    ElseIf (cbMode.ListIndex = 1) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR300
    ElseIf (cbMode.ListIndex = 2) Then
        UFE_SetMode hExtractor, UFE_MODE.SFR300v2

```

```

ElseIf(cbMode.ListIndex = 3) Then
    UFE_SetMode hExtractor, UFE_MODE.General
End If
<sets the template type>
If(cbType.ListIndex = 0) Then
    UFE_SetTemplateType hExtractor,
UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_SUPREMA
ElseIf(cbType.ListIndex = 1) Then
    UFE_SetTemplateType hExtractor,
UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_ISO19794_2
ElseIf(cbType.ListIndex = 2) Then
    UFE_SetTemplateType hExtractor,
UFE_TEMPLATE_TYPE.UFE_TEMPLATE_TYPE_ANSI378
End If
'=====
'=====
'=====

' Create matcher
'=====

ufm_res = UFM_Create(hMatcher)
If(ufm_res = UFM_STATUS.OK) Then
    AddMessage ("UFM_Create: OK" & vbCrLf)
Else
    UFM_GetErrorString ufm_res, m_strError
    AddMessage ("UFM_Create: " & m_strError & vbCrLf)
    GoTo errret
End If
'=====
'=====

' Set Parameters for matcher
'=====

UFM_SetParameter hMatcher, UFM_PARAM.SECURITY_LEVEL, 4
UFM_SetParameter hMatcher, UFM_PARAM.USE_SIF, 0
<template type setting. This must be same with extracted template>
If(cbType.ListIndex = 0) Then
    UFM_SetTemplateType hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_SUPREMA
ElseIf(cbType.ListIndex = 1) Then
    UFM_SetTemplateType hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_ISO19794_2
ElseIf(cbType.ListIndex = 2) Then
    UFM_SetTemplateType hMatcher,
UFM_TEMPLATE_TYPE.UFM_TEMPLATE_TYPE_ANSI378

```

```

End If
=====
====='
<extracts two templates from two loaded images>
If(Not ExtractTemplate(hExtractor, Template1, Template1Size, m_szFilename1)) Then
  GoTo errret
End If
If(Not ExtractTemplate(hExtractor, Template2, Template2Size, m_szFilename2)) Then
  GoTo errret
End If
  <verifies two fingerprint images whether matched or not>
ufm_res = UFM_Verify(hMatcher, Template1(0), Template1Size, Template2(0),
Template2Size, VerifySucceed)
If(ufm_res <> UFM_STATUS.OK) Then
  UFM_GetErrorString ufm_res, m_strError
  AddMessage ("UFM_Verify: " & m_strError & vbCrLf)
  GoTo errret
End If
If(VerifySucceed = 1) Then
  AddMessage ("Verification succeed" & vbCrLf)
Else
  AddMessage ("Verification failed" & vbCrLf)
End If
errret:
=====
====='
' Delete extractor
=====
====='

If(hExtractor <> 0) Then
  ufe_res = UFE_Delete(hExtractor)
  If(ufe_res = UFE_STATUS.OK) Then
    AddMessage ("UFE_Delete: OK" & vbCrLf)
  Else
    UFE_GetErrorString ufe_res, m_strError
    AddMessage ("UFE_Delete: " & m_strError & vbCrLf)
  End If
End If
=====
====='
' Delete matcher
=====
====='

If(hMatcher <> 0) Then
  ufm_res = UFM_Delete(hMatcher)
  If(ufm_res = UFM_STATUS.OK) Then
    AddMessage ("UFM_Delete: OK" & vbCrLf)

```

```

Else
    UFM_GetErrorString ufm_res, m_strError
    AddMessage ("UFM_Delete: " & m_strError & vbCrLf)
End If
End If
'=====
=====
End Sub

```

C#

```

private void btnMatch_Click(object sender, EventArgs e) {
    UFExtractor Extractor = null;
    UFMatcher Matcher = null;
    UFM_STATUS ufm_res;
    byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
    byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
    int Template1Size;
    int Template2Size;
    bool VerifySucceed;
    /////////////////////////////////
    // Create extractor
    /////////////////////////////////
    <creates an extractor>
    Extractor = new UFExtractor();
    /////////////////////////////////
    // Set Parameters and Mode for extractor
    /////////////////////////////////
    <extraction settings. see this page to check 'detect core'>
    Extractor.DetectCore = false;
    Extractor.TemplateSize = MAX_TEMPLATE_SIZE;
    Extractor.UseSIF = false;

    <sets the scanner type>
    switch (cbMode.SelectedIndex) {
        case 0:
            Extractor.Mode = UFE_MODE.SFR200;
            break;
        case 1:
            Extractor.Mode = UFE_MODE.SFR300;
            break;
        case 2:
            Extractor.Mode = UFE_MODE.SFR300v2;
            break;
        case 3:
            Extractor.Mode = UFE_MODE.GENERAL;
            break;
    }
    <sets the template type>

```

```

switch (cbTemplateType.SelectedIndex){
    case 0:
        Extractor.nTemplateType = 2001;
        break;
    case 1:
        Extractor.nTemplateType = 2002;
        break;
    case 2:
        Extractor.nTemplateType = 2003;
        break;
}
/////////////////////////////////////////////////////////////////
// Create matcher
/////////////////////////////////////////////////////////////////
Matcher = new UFMatcher();
/////////////////////////////////////////////////////////////////
// Set Parameters for matcher
/////////////////////////////////////////////////////////////////
Matcher.SecurityLevel = 4;
Matcher.UseSIF = false;
/////////////////////////////////////////////////////////////////
<template type setting. This must be same with extracted template>
switch (cbTemplateType.SelectedIndex)
{
    case 0:
        Matcher.nTemplateType = 2001;
        break;
    case 1:
        Matcher.nTemplateType = 2002;
        break;
    case 2:
        Matcher.nTemplateType = 2003;
        break;
}
<extracts two templates from two loaded images>
if (!ExtractTemplate(ref Extractor, Template1, out Template1Size, m_szFilename1))
{
    return;
}
if (!ExtractTemplate(ref Extractor, Template2, out Template2Size, m_szFilename2))
{
    return;
}
<verifies two fingerprint images whether matched or not>
ufm_res = Matcher.Verify(Template1, Template1Size, Template2, Template2Size,
out VerifySucceed);
if (ufm_res != UFM_STATUS.OK) {

```

```
UFMatcher.GetErrorString(ufm_res, out m_strError);
tbxMessage.AppendText("UFMatcher Verify: " + m_strError + "\r\n");
return;
}
if(VerifySucceed) {
    tbxMessage.AppendText("Verification succeed\r\n");
} else {
    tbxMessage.AppendText("Verification failed\r\n");
}
}
```

UFE30_DatabaseDemo

UFE30_DatabaseDemo provides the demo about managing database. This program uses [UFDatabase](#), [UFScanner](#), and [UFMatcher](#) modules.

Required products

Products	Remarks
Suprema BioMini SDK	No restriction

Available languages

Languages	Locations
Visual C++ 6.0	samples\VS60\UFE30_DatabaseDemoVC60
Visual Basic 6.0	samples\VS60\UFE30_DatabaseDemoVB60
Visual C#	samples\VS80\UFE30_DatabaseDemoCS
Visual Basic .NET	samples\VS80\UFE30_DatabaseDemoVBNET

UFE30_DatabaseDemo_Usage

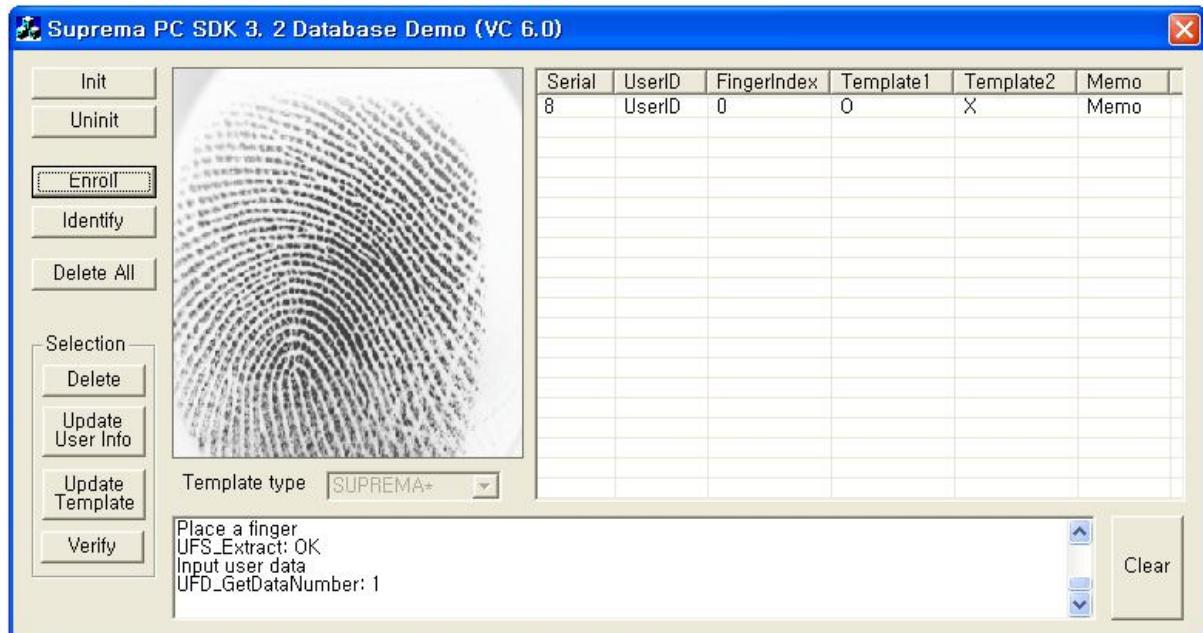
Executable File Location

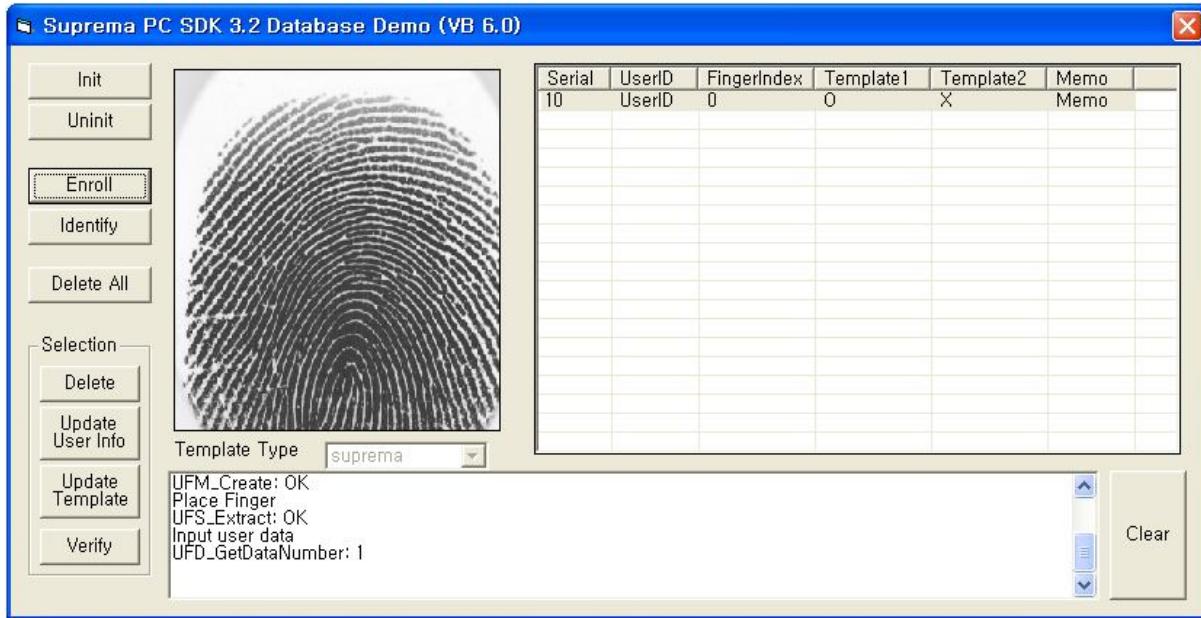
- bin\UFE30_DatabaseDemoVS60.exe
 - bin\UFE30_DatabaseDemoVBNET.exe
 - bin\UFE30_DatabaseDemoCS.exe
 - Source code of all demo application

Required products

Suprema BioMini SDK

Picture of the Demo sample applications





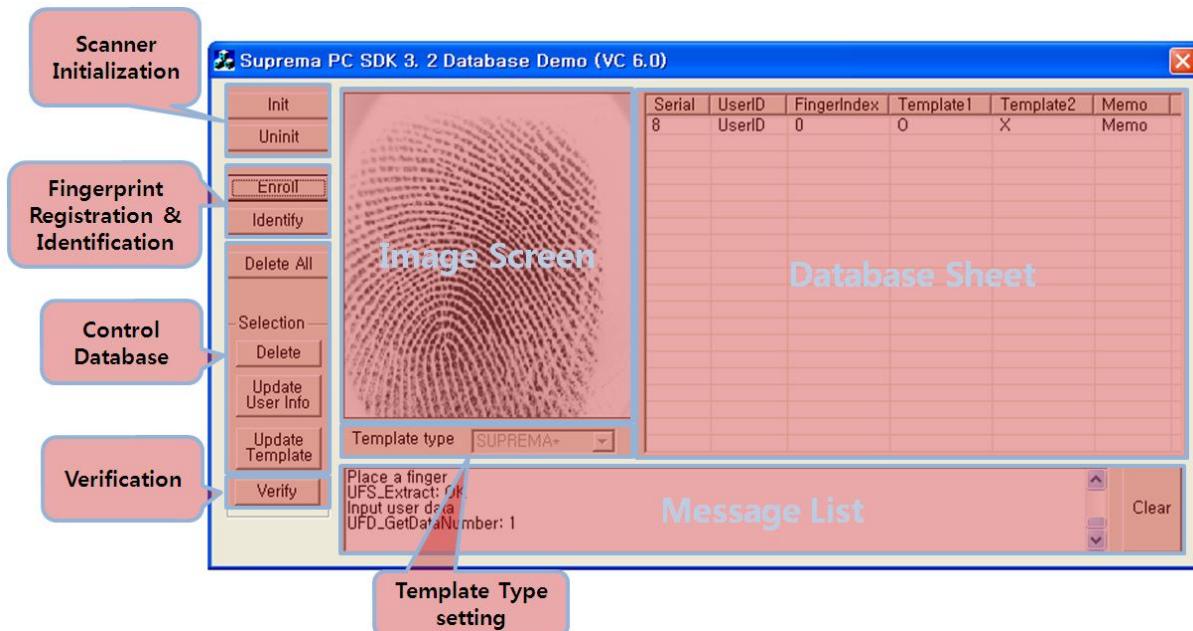
User Interface Components

Suprema PC SDK DatabaseDemo sample demonstrates how to use Suprema PC SDK with MS Access database. Please use this sample as a reference for making your own program. PC SDK Database demo provides following methods::

- Initialize Scanner - A scanner should be initialized for using the functions about the scanner
- Enroll fingerprint - A fingerprint can be enrolled by using BioMini scanner to the database
- Verification - A fingerprint can be verified against selected fingerprint which has been saved in the database
- Identification - A fingerprint can be identified against every enrolled fingerprint which have been saved in the database
- Manage database - Deletes or updates any user data from the database table

Every UFE30_DatabaseDemo application is written by different languages from Visual C++ to C#. But they are made of same functions and same interface design.

The picture bellow shows the PC SDK DatabaseDemo sample main window and description about bunch of interface components :



The table bellow shows the PC SDK DatabaseDemo's simple functionality and links to each detailed usage page :

Scanner initialize	
Items about scanner initialization. Using these components, to makes a connection or disconnection with fingerprint scanner and checking the connection state	
Init button - Initializes UFScanner module	Source Code
Uninit button - Uninitializes UFScanner module	Source Code

Database control	
Items for controlling database	
Delete All button - Removes all input data from the database	Source Code
Delete button - Removes a selected data from the database	Source Code
Update User Info button - Updates a selected user information data in the database	Source Code
Update Template button - Updates a selected template data in the database	Source Code

Enroll	
Extracts a template from captured image. And inserts the template data and user data into database	
Enroll button - Enrolls a fingerprint into the database	Source Code

Match	
Items for verification and identification	
Identify button - Identifies fingerprint from enrolled fingerprints	Source Code
Verify button - Verifies input fingerprint whether matched or not compared with selected finger from the database dialog	Source Code

Initialization with Database Sample

Description

To capture a fingerprint image, one or more scanner has to be initialized with your program. This page is the source code about initialization task that is connected with 'Init' button in 'database' samples.

Using functions

[UFS_Init](#), [UFS_GetErrorString](#), [UFS_GetScannerNumber](#), [UFS_GetScannerHandle](#),
[UFD_Open](#), [UFD_GetErrorString](#), [UFM_Create](#), [UFM_GetErrorString](#)

wrapper

Source Code

VS60

```
void CUFE30_DatabaseDemoDlg::OnInit()
{
    //////////////// Initialize scanner module ///////////////////
    // Initilize scanner module
    //////////////// Initialize scanner module ///////////////////
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    int nScannerNumber;
    BeginWaitCursor();
    <scanner initialization function
    see UFS\_Init to check the return values. This function returns true value if the
    initialization success, false otherwise>
    ufs_res = UFS_Init();
    EndWaitCursor();
    if (ufs_res == UFS_OK) {
        AddMessage("UFS_Init: OK\r\n");
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFS_GetErrorString(ufs_res, m_strError);
        AddMessage("UFS_Init: %s\r\n", m_strError);
        return;
    }
    <gets the number of scanners>
    ufs_res = UFS_GetScannerNumber(&nScannerNumber);
    if (ufs_res == UFS_OK) {
        AddMessage("UFS_GetScannerNumber: %d\r\n", nScannerNumber);
    }
}
```

```

} else {
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_GetScannerNumber: %s\r\n", m_strError);
    return;
}
if(nScannerNumber == 0) {
    AddMessage("There's no available scanner\r\n");
    return;
} else {
    AddMessage("First scanner will be used\r\n");
}
<uses the first scanner>
<gets the first(No.0) scanner handle>
ufs_res = UFS_GetScannerHandle(0, &m_hScanner);
if(ufs_res != UFS_OK) {
    UFS_GetErrorString(ufs_res, m_strError);
    AddMessage("UFS_GetScannerHandle: %s\r\n", m_strError);
    return;
}
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
// Open database
////////////////////////////////////////////////////////////////////////
<ufd_res takes UFDatabase function's return value>
UFD_STATUS ufd_res;
// Generate connection string
CString szDataSource;
CString szConnection;
/*
szDataSource = "UFDatabase.mdb";
*/
AddMessage("Select a database file\r\n");
szDataSource = "UFDatabase.mdb";
<makes a filedialog to get a database file location>
CFileDialog dlg(FALSE, "mdb", szDataSource, NULL, "Database Files (*.mdb)|*.mdb");
if(dlg.DoModal() != IDOK) {
    AddMessage("DB selection is cancelled by user\r\n");
    return;
}
szDataSource = dlg.GetPathName();
szConnection.Format(TEXT("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=%s;"), szDataSource);
<opens a database file. Please see this page>
ufd_res = UFD_Open(szConnection, NULL, NULL, &m_hDatabase);
if(ufd_res == UFD_OK) {
    AddMessage("UFD_Open: OK\r\n");
} else {

```

```

<gets an error message string according to ufd_res which is a return value of
UFD function>
    UFD_GetErrorString(ufd_res, m_strError);
    AddMessage("UFD_Open: %s\r\n", m_strError);
    return;
}

UpdateDatabaseList();
///////////////////////////////
// Create matcher
/////////////////////////////
<creates a matcher>
    UFM_STATUS ufm_res;
    ufm_res = UFM_Create(&m_hMatcher);
    if(ufm_res == UFM_OK) {
        AddMessage("UFM_Create: OK\r\n");
    } else {
        UFM_GetErrorString(ufm_res, m_strError);
        AddMessage("UFM_Create: %s\r\n", m_strError);
        return;
    }
/////////////////////////////
}

```

VB60

```

Private Sub btnInit_Click()
'=====
' Initialize scanners
'=====

<ufs_res takes UFSanner function's return value>
Dim ufs_res As UFS_STATUS
Dim ScannerNumber As Long

Screen.MousePointer = vbHourglass
ufs_res = UFS_Init()
Screen.MousePointer = vbDefault
If(ufs_res = UFS_STATUS.OK) Then
    AddMessage ("UFS_Init: OK" & vbCrLf)
Else
    <gets an error message string according to ufs_res which is a return value of
    UFS function>
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_Init: " & m_strError & vbCrLf)
    Exit Sub
End If

<gets the number of scanners>

```

```

ufs_res = UFS_GetScannerNumber(ScannerNumber)
If (ufs_res = UFS_STATUS.OK) Then
    AddMessage ("UFS_GetScannerNumber: " & ScannerNumber & vbCrLf)
Else
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_GetScannerNumber: " & m_strError & vbCrLf)
    Exit Sub
End If

If (ScannerNumber = 0) Then
    AddMessage ("There's no available scanner" & vbCrLf)
    Exit Sub
Else
    AddMessage ("First scanner will be used" & vbCrLf)
End If
<uses the first scanner>
    <gets the first(No.0) scanner handle>
ufs_res = UFS_GetScannerHandle(0, m_hScanner)
If (ufs_res <> UFS_STATUS.OK) Then
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_GetScannerHandle: " & m_strError & vbCrLf)
    Exit Sub
End If
'-----
'-----
'-----
' Open database
'-----
'<ufd_res takes UFDatabase function's return value>
Dim ufd_res As UFD_STATUS

Dim Connection As String
Dim DataSource As String

DataSource = "UFDatabase.mdb"
<makes a filedialog to get a database file location>
cdFileDialog.Filter = "Database Files (*.mdb)|*.mdb"
cdFileDialog.FileName = "UFDatabase.mdb"
cdFileDialog.DefaultExt = "mdb"
cdFileDialog.Flags = cdlOFNHideReadOnly Or cdlOFNPathMustExist Or
cdlOFNOverwritePrompt Or cdlOFNNoReadOnlyReturn
cdFileDialog.ShowOpen
DataSource = cdFileDialog.FileName

Connection = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & DataSource & ";"
```

```

<opens a database file. Please see this page>
ufd_res = UFD_Open(Connection, vbNullString, vbNullString, m_hDatabase)
If (ufd_res = UFD_STATUS.OK) Then
    AddMessage ("UFD_Open: OK" & vbCrLf)
Else
    <gets an error message string according to ufd_res which is a return value of
     UFD function>
    UFD_GetErrorString ufd_res, m_strError
    AddMessage ("UFD_Open: " & m_strError & vbCrLf)
End If

UpdateDatabaseList
'=====
'=====

'=====
'=====

' Create matcher
'=====
'=====

    <creates a matcher>
Dim ufm_res As UFM_STATUS
ufm_res = UFM_Create(m_hMatcher)
If (ufm_res = UFM_STATUS.OK) Then
    AddMessage ("UFM_Create: OK" & vbCrLf)
Else
    UFM_GetErrorString ufm_res, m_strError
    AddMessage ("UFM_Create: " & m_strError & vbCrLf)
    Exit Sub
End If
'=====
'=====

End Sub

```

C#

```

private void btnInit_Click(object sender, EventArgs e) {
    //=====
    //=====
    // Initialize scanners
    //=====
    //=====

    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;
    int nScannerNumber;
    Cursor.Current = Cursors.WaitCursor;
    ufs_res = m_ScannerManager.Init();
    Cursor.Current = this.Cursor;
    if (ufs_res == UFS_STATUS.OK) {

```

```

        tbxMessage.AppendText("UFSscanner Init: OK\r\n");
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFSscanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFSscanner Init: " + m_strError + "\r\n");
        return;
    }
    <gets the number of scanners>
    nScannerNumber = m_ScannerManager.Scanners.Count;
    tbxMessage.AppendText("UFSscanner GetScannerNumber: " + nScannerNumber +
    "\r\n");
    if(nScannerNumber == 0) {
        tbxMessage.AppendText("There's no available scanner\r\n");
        return;
    } else {
        tbxMessage.AppendText("First scanner will be used\r\n");
    }
    <uses the first scanner>
    <gets the first(No.0) scanner handle>
    m_Scanner = m_ScannerManager.Scanners[0];
    //=====
    =====//
    //=====
    =====//
    // Open database
    //=====
    =====//
    <ufd_res takes UFDatabase function's return value>
    UFD_STATUS ufd_res;
    m_Database = new UFDatabase();

    // Generate connection string
    string szDataSource;
    string szConnection;
    /*
    szDataSource = "UFDatabase.mdb";
    */
    tbxMessage.AppendText("Select a database file\r\n");
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.FileName = "UFDatabase.mdb";
    dlg.Filter = "Database Files (*.mdb)|*.mdb";
    dlg.DefaultExt = "mdb";
    <makes a filedialog to get a database file location>
    DialogResult res = dlg.ShowDialog();
    if(res != DialogResult.OK) {
        return;
    }
    szDataSource = dlg.FileName;

```

```
szConnection = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + szDataSource
+ ";";
<opens a database file. Please see this page>
ufd_res = m_Database.Open(szConnection, null, null);
if(ufd_res == UFD_STATUS.OK) {
    tbxMessage.AppendText("UFDatabase Open: OK\r\n");
} else {
    <gets an error message string according to ufd_res which is a return value of
    UFD function>
    UFDatabase.GetErrorString(ufd_res, out m_strError);
    tbxMessage.AppendText("UFDatabase Open: " + m_strError + "\r\n");
    return;
}
UpdateDatabaseList();
//=====
=====//
//=====
=====//
// Create matcher
//=====
=====//
<creates a matcher>
m_Matcher = new UFMatcher();
//=====
=====//
}
```

Uninitialization

Description

Database demo's 'Uninit' button is set to calling the OnUninit function. This function releases the scanner resources and closes the opened database.

Using functions

[UFS_Uninit](#), [UFS_GetErrorString](#), [UFD_Close](#), [UFM_Delete](#),
[UFD_GetErrorString](#), [UFM_GetErrorString](#)

Source Code

VS60
<pre>void CUFE30_DatabaseDemoDlg::OnUninit() { /// // Uninit scanner module /// <ufs_res takes UFSanner function's return value> UFS_STATUS ufs_res; BeginWaitCursor(); <uninitializes a scanner, use this to release resources which are related to the scanner. See UFS_Uninit to check the return value> ufs_res = UFS_Uninit(); EndWaitCursor(); if(ufs_res == UFS_OK) { AddMessage("UFS_Uninit: OK\r\n"); } else { <gets an error message string according to ufs_res which is a return value of UFS function> UFS_GetErrorString(ufs_res, m_strError); AddMessage("UFS_Uninit: %s\r\n", m_strError); } m_hScanner = NULL; /// /// // Close database /// UFD_STATUS ufd_res; if(m_hDatabase != NULL) { <closes the database> ufd_res = UFD_Close(m_hDatabase); if(ufd_res == UFD_OK) { }</pre>

```

        AddMessage("UFD_Close: OK\r\n");
    } else {
        UFD_GetErrorString(ufd_res, m_strError);
        AddMessage("UFD_Close: %s\r\n", m_strError);
    }
    m_hDatabase = NULL;
}
m_ctlDatabaseList.DeleteAllItems();
// Delete matcher
// Delete matcher
if(m_hMatcher != NULL) {
    UFM_STATUS ufm_res;
    ufm_res = UFM_Delete(m_hMatcher);
    if(ufm_res == UFM_OK) {
        AddMessage("UFM_Delete: OK\r\n");
    } else {
        UFM_GetErrorString(ufm_res, m_strError);
        AddMessage("UFM_Delete: %s\r\n", m_strError);
    }
    m_hMatcher = NULL;
}
// Delete matcher
}

```

VB60

```

Private Sub btnUninit_Click()
'-----
'-----'
' Uninit scanners
'-----'
'-----'
<ufs_res takes UFSanner function's return value>
Dim ufs_res As UFM_STATUS

Screen.MousePointer = vbHourglass
    <uninitializes a scanner, use this to release resources which are related to the scanner.
    See UFS\_Uninit to check the return value>
ufs_res = UFS_Uninit()
Screen.MousePointer = vbDefault
If (ufs_res = UFS_STATUS.OK) Then
    AddMessage ("UFS_Uninit: OK" & vbCrLf)
Else
    <gets an error message string according to ufs_res which is a return value of
    UFS function>
    UFS_GetErrorString ufs_res, m_strError
    AddMessage ("UFS_Uninit: " & m_strError & vbCrLf)
End If

```

```

'====='
'====='
' Close database
'====='

If(m_hDatabase <> 0) Then
    Dim ufd_res As UFD_STATUS

        <closes the database>
    ufd_res = UFD_Close(m_hDatabase)
    If(ufd_res = UFD_STATUS.OK) Then
        AddMessage ("UFD_Close: OK" & vbCrLf)
    Else
        UFD_GetErrorString ufd_res, m_strError
        AddMessage ("UFD_Close: " & m_strError & vbCrLf)
    End If
    m_hDatabase = 0
End If

lvDatabaseList.ListItems.Clear
'====='
'====='
' Delete matcher
'====='

If(m_hMatcher <> 0) Then
    Dim ufm_res As UFM_STATUS
    ufm_res = UFM_Delete(m_hMatcher)
    If(ufm_res = UFM_STATUS.OK) Then
        AddMessage ("UFM_Delete: OK" & vbCrLf)
    Else
        UFM_GetErrorString ufm_res, m_strError
        AddMessage ("UFM_Delete: " & m_strError & vbCrLf)
    End If
    m_hMatcher = 0
End If

'====='
'====='
End Sub

```

C#

```

private void btnUninit_Click(object sender, EventArgs e) {
    //=====
    //=====
    // Uninit scanner module
    //=====
    //=====
    <ufs_res takes UFSanner function's return value>
    UFS_STATUS ufs_res;

    Cursor.Current = Cursors.WaitCursor;
    <uninitializes a scanner, use this to release resources which are related to the scanner.
    See UFS\_Uninit to check the return value>
    ufs_res = m_ScannerManager.Uninit();
    Cursor.Current = this.Cursor;
    if(ufs_res == UFS_STATUS.OK) {
        tbxMessage.AppendText("UFScanner Uninit: OK\r\n");
    } else {
        <gets an error message string according to ufs_res which is a return value of
        UFS function>
        UFScanner.GetErrorString(ufs_res, out m_strError);
        tbxMessage.AppendText("UFScanner Uninit: " + m_strError + "\r\n");
    }
    //=====
    //=====

    //=====
    //=====
    // Close database
    //=====
    //=====

    UFD_STATUS ufd_res;
    if(m_Database != null) {
        <closes the database>
        ufd_res = m_Database.Close();
        if(ufd_res == UFD_STATUS.OK) {
            tbxMessage.AppendText("UFDDatabase Close: OK\r\n");
        } else {
            UFDDatabase.GetErrorString(ufd_res, out m_strError);
            tbxMessage.AppendText("UFDDatabase Close: " + m_strError + "\r\n");
        }
    }
    lvDatabaseList.Items.Clear();
    //=====
    //=====

}

```

Enrollment with Database

Description

Database demo sample's enroll function captures a fingerprint image and extracts a template from the fingerprint image by using ExtractTemplate(VS60) function. And inserts the template data with other user data into the database by using UFD_AddData function.

Using functions

[UFS_SetTemplateType](#), [UFS_ClearCaptureImageBuffer](#) in ExtracTemplate,
[UFS_CaptureSingleImage](#) in ExtracTemplate, [UFS_Extract](#) in ExtracTemplate,
[UFS_GetErrorString](#), [UFD_AddData](#), [UFD_GetErrorString](#)

Source Code

VS60
<pre> void CUFE30_DatabaseDemoDlg::OnEnroll() { switch (m_nType) { case 0: UFS_SetTemplateType(m_hScanner, UFS_TEMPLATE_TYPE_SUPREMA); AddMessage("Current scanner,UFS_SetTemplateType: SUPREMA TYPE \r\n"); break; case 1: UFS_SetTemplateType(m_hScanner, UFS_TEMPLATE_TYPE_ISO19794_2); AddMessage("Current scanner,UFS_SetTemplateType: ISO_19794_2 TYPE \r\n"); break; case 2: UFS_SetTemplateType(m_hScanner, UFS_TEMPLATE_TYPE_ANSI378); AddMessage("Current scanner,UFS_SetTemplateType: ANSI378 TYPE \r\n"); break; } <extracts a template after capturing a fingerprint. The extracted template data will be saved in the Template variable> if (!ExtractTemplate(m_Template1, &m_nTemplate1Size)) { return; } Invalidate(FALSE); ///////////////////////////////// <gets data from Userinfo dialog> CUserInfoDlg dlg;</pre>

```

    UFD_STATUS ufd_res;
    AddMessage("Input user data\r\n");
    if(dlg.DoModal() != IDOK) {
        AddMessage("User data input is cancelled by user\r\n");
        return;
    }
    <inserts data into the database>
    ufd_res = UFD_AddData(m_hDatabase, dlg.m_strUserID, dlg.m_nFingerIndex,
    m_Template1, m_nTemplate1Size, NULL, 0, dlg.m_strMemo);
    if(ufd_res != UFD_OK) {
        UFD_GetErrorString(ufd_res, m_strError);
        AddMessage("UFD_AddData: %s\r\n", m_strError);
        ((CComboBox*)GetDlgItem(IDC_TYPE))->EnableWindow(FALSE);
    }

    UpdateDatabaseList();
    /////////////////////////////////
}

```

VB60

```

Private Sub btnEnroll_Click()
    If(cbType.ListIndex = 0) Then
        UFS_SetTemplateType m_hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_SUPREMA
    ElseIf(cbType.ListIndex = 1) Then
        UFS_SetTemplateType m_hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ISO19794_2
    ElseIf(cbType.ListIndex = 2) Then
        UFS_SetTemplateType m_hScanner,
        UFS_TEMPLATE_TYPE.UFS_TEMPLATE_TYPE_ANSI378
    End If

    <extracts a template after capturing a fingerprint. The extracted template data will be
    saved in the Template variable>
    If(Not ExtractTemplate(m_Template1, m_Template1Size)) Then
        Exit Sub
    End If

    Dim ufd_res As UFD_STATUS
    <gets data from UserInfo dialog>
    AddMessage ("Input user data" & vbCrLf)
    UserInfoForm.SetAdd
    UserInfoForm.Show vbModal
    If(Not UserInfoForm.DialogResult) Then
        AddMessage ("User data input is cancelled by user" & vbCrLf)
        Exit Sub
    Else
        pbImageFrame.Refresh
    End If

```

```

<inserts data into the database>
    ufd_res = UFD_AddData(m_hDatabase, UserInfoForm.txtUserID,
    Val(UserInfoForm.txtFingerIndex), m_Template1(0), m_Template1Size, vbNull, 0,
    UserInfoForm.txtMemo)
    If (ufd_res <> UFD_STATUS.OK) Then
        UFD_GetErrorString ufd_res, m_strError
        AddMessage ("UFD_AddData: " & m_strError & vbCrLf)
    Else
        cbType.Enabled = False
        cbType.Locked = True
    End If

    UpdateDatabaseL
End Sub

```

C#

```

private void btnEnroll_Click(object sender, EventArgs e) {
    <extracts a template after capturing a fingerprint. The extracted template data will be
    saved in the Template variable>
    if (!ExtractTemplate(m_Template1, out m_Template1Size)) {
        return;
    }
    DrawCapturedImage(m_Scanner);
    <gets data from Userinfo dialog>
    UserInfoForm dlg = new UserInfoForm(false);
    UFD_STATUS ufd_res;
    tbxMessage.AppendText("Input user data\r\n");
    if (dlg.ShowDialog(this) != DialogResult.OK) {
        tbxMessage.AppendText("User data input is cancelled by user\r\n");
        return;
    }
    <inserts data into the database>
    ufd_res = m_Database.AddData(dlg.UserID, dlg.FingerIndex, m_Template1,
    m_Template1Size, null, 0, dlg.Memo);
    if (ufd_res != UFD_STATUS.OK)
    {
        UFDatabase.GetErrorString(ufd_res, out m_strError);
        tbxMessage.AppendText("UFDatabase AddData: " + m_strError + "\r\n");
    } else {

```

```
        cbScanTemplateType.Enabled = false;  
    }  
    UpdateDatabaseList();  
}
```

Identification with Database

Description

Database demo sample's 'Identify' button is set to calling the OnIdentify(VS60) function. To identify a fingerprint in the database sample, whole enrolled template data should be loaded from the database by using UFD_GetTemplateNumber and UFD_GetTemplateListWithSerial function. And then extracts a template from user's finger by calling ExtractTemplate(VS60) function. And calls the UFM_Identify function to execute 1:N matching using the extracted template and all the other loaded templates.

Using functions

[UFD_GetTemplateNumber](#), [UFD_GetTemplateListWithSerial](#),
[UFS_ClearCaptureImageBuffer](#) in ExtracTemplate, [UFS_CaptureSingleImage](#) in
 ExtracTemplate, [UFS_Extract](#) in ExtracTemplate, [UFS_GetErrorString](#), [UFM_Identify](#),
[UFM_GetErrorString](#), [UFD_GetErrorString](#)

Source Code

VS60
<pre>void CUFE30_DatabaseDemoDlg::OnIdentify() { <ufd_res takes UFDatabase function's return value> UFD_STATUS ufd_res; <ufm_res takes UFMatcher function's return value> UFM_STATUS ufm_res; // Input finger data unsigned char Template[MAX_TEMPLATE_SIZE]; int nTemplateSize; // DB data unsigned char** ppDBTemplate = NULL; int* pnDBTemplateSize = NULL; int* pnDBSerial = NULL; int nDBTemplateNum; int nMatchIndex; int i; <gets the number of template. Please see UFD_GetTemplateNumber> ufd_res = UFD_GetTemplateNumber(m_hDatabase, &nDBTemplateNum); if(ufd_res != UFD_OK) { <gets an error message string according to ufd_res which is a return value of UFD function> UFD_GetErrorString(ufd_res, m_strError); AddMessage("UFD_GetTemplateNumber: %s\r\n", m_strError); return; } }</pre>

```

}

/////////////////////////////
// Allocate DB data
/////////////////////////////
ppDBTemplate = (unsigned char**)malloc(nDBTemplateNum * sizeof(unsigned
char**));
pnDBTemplateSize = (int*)malloc(nDBTemplateNum * sizeof(int*));
pnDBSerial = (int*)malloc(nDBTemplateNum * sizeof(int*));
for (i = 0; i < nDBTemplateNum; i++) {
    ppDBTemplate[i] = (unsigned char*)malloc(MAX_TEMPLATE_SIZE *
sizeof(unsigned char*));
}
/////////////////////////////
<gets all template list using serial number. Please see this page>
ufd_res = UFD_GetTemplateListWithSerial(m_hDatabase, ppDBTemplate,
pnDBTemplateSize, pnDBSerial);
if (ufd_res != UFD_OK) {
    UFD_GetErrorString(ufd_res, m_strError);
    AddMessage("UFD_GetTemplateListWithSerial: %s\r\n", m_strError);
    goto errret;
}
<extracts a template after capturing a fingerprint. The extracted template data will be
saved in the Template variable>
if (!ExtractTemplate(Template, &nTemplateSize)) {
    goto errret;
}
<identifies a fingerprint template against all saved fingerprint templates>
ufm_res = UFM_Identify(m_hMatcher, Template, nTemplateSize, ppDBTemplate,
pnDBTemplateSize, nDBTemplateNum, 5000, &nMatchIndex);
if (ufm_res != UFD_OK) {
    UFM_GetErrorString(ufm_res, m_strError);
    AddMessage("UFM_Identify: %s\r\n", m_strError);
    return;
}
if (nMatchIndex != -1) {
    AddMessage("Identification succeed (Serial = %d)\r\n",
pnDBSerial[nMatchIndex]);
} else {
    AddMessage("Identification failed\r\n");
}
errret:
/////////////////////////////
// Free DB data
/////////////////////////////
if (ppDBTemplate != NULL) {
    for (i = 0; i < nDBTemplateNum; i++) {
        free(ppDBTemplate[i]);
    }
}

```

```

        free(ppDBTemplate);
    }
    if(pnDBTemplateSize != NULL) {
        free(pnDBTemplateSize);
    }
    if(pnDBSerial != NULL) {
        free(pnDBSerial);
    }
    ///////////////////////////////////////////////////
}

```

VB60

```

Private Sub btnIdentify_Click()
    <ufd_res takes UFDdatabase function's return value>
    Dim ufd_res As UFD_STATUS
    <ufm_res takes UFMatcher function's return value>
    Dim ufm_res As UFM_STATUS
    ' Input finger data
    Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
    Dim TemplateSize As Long
    ' DB data
    Dim DBTemplate() As Byte
    Dim DBTemplateSize() As Long
    Dim DBSerial() As Long
    Dim DBTemplatePtr() As Long
    Dim DBTemplateNum As Long
    Dim MatchIndex As Long

    <gets the number of template. Please see UFD\_GetTemplateNumber>
    ufd_res = UFD_GetTemplateNumber(m_hDatabase, DBTemplateNum)
    If(ufd_res = UFD_STATUS.OK) Then
        AddMessage ("UFD_GetTemplateNumber: " & DBTemplateNum & vbCrLf)
    Else
        <gets an error message string according to ufd_res which is a return value of
        UFD function>
        UFD_GetErrorString ufd_res, m_strError
        AddMessage ("UFD_GetTemplateNumber: " & m_strError & vbCrLf)
    End If

    ReDim DBTemplate(MAX_TEMPLATE_SIZE - 1, DBTemplateNum - 1) As Byte
    ReDim DBTemplateSize(DBTemplateNum - 1) As Long
    ReDim DBSerial(DBTemplateNum - 1) As Long

    ' Make template pointer array to pass two dimensional template data
    ReDim DBTemplatePtr(DBTemplateNum - 1) As Long
    Dim i As Long
    For i = 0 To DBTemplateNum - 1
        DBTemplatePtr(i) = VarPtr(DBTemplate(0, i))
    Next

```

```

    <gets all template list using serial number, see this page>
    ufd_res = UFD_GetTemplateListWithSerial(m_hDatabase, DBTemplatePtr(0),
DBTemplateSize(0), DBSerial(0))
    If(ufd_res <> UFD_STATUS.OK) Then
        UFD_GetErrorString ufd_res, m_strError
        AddMessage ("UFD_GetTemplateListWithSerial: " & m_strError & vbCrLf)
        Exit Sub
    End If

    <extracts a template after capturing a fingerprint. The extracted template data will be
    saved in the Template variable>
    If(Not ExtractTemplate(Template, TemplateSize)) Then
        Exit Sub
    Else
        pbImageFrame.Refresh
    End If

    Screen.MousePointer = vbHourglass
    <identifies a fingerprint template against all saved fingerprint templates>
    ufm_res = UFM_Identify(m_hMatcher, Template(0), TemplateSize, DBTemplatePtr(0),
DBTemplateSize(0), DBTemplateNum, 5000, MatchIndex)
    Screen.MousePointer = vbDefault
    If(ufm_res <> UFM_STATUS.OK) Then      UFM_GetErrorString ufm_res,
m_strError
        AddMessage ("UFM_Identify: " & m_strError & vbCrLf)
        Exit Sub
    End If

    If(MatchIndex <> -1) Then
        AddMessage ("Identification succeed (Serial = " & DBSerial(MatchIndex) & ")" &
vbCrLf)
    Else
        AddMessage ("Identification failed" & vbCrLf)
    End If
End Sub

```

C#

```

private void btnIdentify_Click(object sender, EventArgs e) {
    <ufd_res takes UFDatabase function's return value>
    UFD_STATUS ufd_res;
    <ufm_res takes UFMatcher function's return value>
    UFM_STATUS ufm_res;
    // Input finger data
    byte[] Template = new byte[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    // DB data
    byte[][] DBTemplate = null;
}

```

```

int [] DBTemplateSize = null;
int [] DBSerial = null;
int DBTemplateNum;
//
int MatchIndex;
<gets all template list using a serial number. Please see this page>
ufd_res = m_Database.GetTemplateListWithSerial(out DBTemplate, out
DBTemplateSize, out DBTemplateNum, out DBSerial);
if(ufd_res != UFD_STATUS.OK) {
    <gets an error message string according to ufd_res which is a return value of
    UFD function>
    UFDatabase.GetErrorString(ufd_res, out m_strError);
    tbxMessage.AppendText("UFD_GetTemplateListWithSerial: " + m_strError +
    "\r\n");
    return;
}
<extracts a template after capturing a fingerprint. The extracted template data will be
saved in the Template variable>
if(!ExtractTemplate(Template, out TemplateSize)) {
    return;
}
DrawCapturedImage(m_Scanner);

Cursor.Current = Cursors.WaitCursor;
<identifies a fingerprint template against all saved fingerprint templates>
ufm_res = m_Matcher.Identify(Template, TemplateSize, DBTemplate,
DBTemplateSize, DBTemplateNum, 5000, out MatchIndex);
Cursor.Current = this.Cursor;
if(ufm_res != UFM_STATUS.OK) {
    UFMatcher.GetErrorString(ufm_res, out m_strError);
    tbxMessage.AppendText("UFMatcher Identify: " + m_strError + "\r\n");
    return;
}
if(MatchIndex != -1) {
    tbxMessage.AppendText("Identification succeed (Serial = " +
    DBSerial[MatchIndex] + ") \r\n");
} else {
    tbxMessage.AppendText("Identification failed\r\n");
}
}
}

```

Verification with Database

Description

Database demo sample's 'Verify' button is set to calling OnSelectionVerify function. This function gets a template data which is selected at the database dialog by using UFD_GetDataBySerial function. And then extracts a template from user's finger by calling ExtractTemplate(VS60) function. And the UFM_Verify function executes 1:1 matching using the extracted template and the selected template.

Using functions

[UFD_GetDataBySerial](#), [UFS_ClearCaptureImageBuffer](#) in ExtractTemplate,
[UFS_CaptureSingleImage](#) in ExtractTemplate, [UFS_Extract](#) in ExtractTemplate,
[UFS_GetErrorString](#), [UFM_Verify](#), [UFM_GetErrorString](#), [UFD_GetErrorString](#)

Source Code

VS60

```
void CUFE30_DatabaseDemoDlg::OnSelectionVerify()
{
    <ufs_res takes UFSanner function's return value>
    UFD_STATUS ufd_res;
    <ufm_res takes UFMatcher function's return value>
    UFM_STATUS ufm_res;
    int nSelected;
    int nSerial;
    // Input finger data
    unsigned char Template[MAX_TEMPLATE_SIZE];
    int nTemplateSize;

    int bVerifySucceed;
    nSelected = m_ctlDatabaseList.GetSelectionMark();
    if (nSelected == -1) {
        AddMessage("Select data\r\n");
        return;
    } else {
        nSerial = atoi(m_ctlDatabaseList.GetItemText(nSelected,
            DATABASE_COL_SERIAL));
    }
    <gets a template data with corresponding serial number. Please see this page>
    ufd_res = UFD_GetDataBySerial(m_hDatabase, nSerial,
        m_szUserID, &m_nFingerIndex, m_Template1, &m_nTemplate1Size,
        m_Template2, &m_nTemplate2Size, m_szMemo);
```

```

if(ufd_res != UFD_OK) {
    <gets an error message string according to the ufd_res which is a return value
     for UFD function>
    UFD_GetErrorString(ufd_res, m_strError);
    AddMessage("UFD_UpdateDataBySerial: %s\r\n", m_strError);
    return;
}
<extracts a template after capturing a fingerprint. The extracted template data will be
 saved in the 'Template' variable>
if(!ExtractTemplate(Template, &nTemplateSize)) {
    return;
}
<verifies two fingerprint templates whether matched or not>
ufm_res = UFM_Verify(m_hMatcher, Template, nTemplateSize, m_Template1,
m_nTemplate1Size, &bVerifySucceed);
if(ufm_res != UFD_OK) {
    UFM_GetErrorString(ufm_res, m_strError);
    AddMessage("UFM_Verify: %s\r\n", m_strError);
    return;
}
if(bVerifySucceed) {
    AddMessage("Verification succeed (Serial = %d)\r\n", nSerial);
} else {
    AddMessage("Verification failed\r\n");
}
}
}

```

VB60

```

Private Sub btnSelectionVerify_Click()
    <ufd_res takes UFDaScanner function's return value>
    Dim ufd_res As UFD_STATUS
    <ufm_res takes UFMatcher function's return value>
    Dim ufm_res As UFM_STATUS
    Dim Serial As Long
    ' Input finger data
    Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
    Dim TemplateSize As Long
    Dim VerifySucceed As Long

    If (Not lvDatabaseList.SelectedItem.Selected) Then
        AddMessage ("Select data" & vbCrLf)
        Exit Sub
    Else
        Serial = Val(lvDatabaseList.SelectedItem.text)
    End If

    <gets a template data with corresponding serial number. Please see this page>
    ufd_res = UFD_GetDataBySerial(m_hDatabase, Serial, m_UserID, m_FingerIndex,
m_Template1(0), m_Template1Size, m_Template2(0), m_Template2Size, m_Memo)

```

```

If(ufd_res <> UFD_STATUS.OK) Then

    <gets an error message string according to the ufd_res which is a return value
     for UFD function>
    UFD_GetErrorString ufd_res, m_strError
    AddMessage ("UFD_GetDataBySerial: " & m_strError & vbCrLf)
    Exit Sub
End If

    <extracts a template after capturing a fingerprint. The extracted template data will be
     saved in the 'Template' variable>
If(Not ExtractTemplate(Template, TemplateSize)) Then
    Exit Sub
Else
    pbImageFrame.Refresh
End If

    <verifies two fingerprint templates whether matched or not>
    ufm_res = UFM_Verify(m_hMatcher, Template(0), TemplateSize, m_Template1(0),
    m_Template1Size, VerifySucceed)
    If(ufm_res <> UFM_STATUS.OK) Then
        UFM_GetErrorString ufm_res, m_strError
        AddMessage ("UFM_Verify: " & m_strError & vbCrLf)
        Exit Sub
    End If
    If(VerifySucceed = 1) Then
        AddMessage ("Verification succeed (Serial = " & Serial & ")" & vbCrLf)
    Else
        AddMessage ("Verification failed" & vbCrLf)
    End If
End Sub

```

C#

```

private void btnSelectionVerify_Click(object sender, EventArgs e) {
    <ufd_res takes UFDaSanner function's return value>
    UFD_STATUS ufd_res;
    <ufm_res takes UFMatcher function's return value>
    UFM_STATUS ufm_res;
    int Serial;
    // Input finger data
    byte[] Template = new byte[MAX_TEMPLATE_SIZE];
    int TemplateSize;
    //
    bool VerifySucceed;
    if(lvDatabaseList.SelectedIndices.Count == 0) {
        tbxMessage.AppendText("Select data\r\n");
        return;
    } else {

```

```

Serial =
    Convert.ToInt32(lvDatabaseList.SelectedItems[0].SubItems[DATABASE_CO
L_SERIAL].Text);
}

<gets a template data with corresponding serial number. Please see this page>
ufd_res = m_Database.GetDataBySerial(Serial,
    out m_UserID, out m_FingerIndex, m_Template1, out m_Template1Size,
    m_Template2, out m_Template2Size, out m_Memo);
if(ufd_res != UFD_STATUS.OK) {
    <gets an error message string according to the ufd_res which is a return value
    for UFD function>
    UFDatabase.GetErrorString(ufd_res, out m_strError);
    tbxMessage.AppendText("UFDatabase UpdateDataBySerial: " + m_strError +
        "\r\n");
    return;
}
<extracts a template after capturing a fingerprint. The extracted template data will be
saved in the 'Template' variable>
if(!ExtractTemplate(Template, out TemplateSize)) {
    return;
}

DrawCapturedImage(m_Scanner);
<verifies two fingerprint templates whether matched or not>
ufm_res = m_Matcher.Verify(Template, TemplateSize, m_Template1,
    m_Template1Size, out VerifySucceed);
if(ufm_res != UFM_STATUS.OK) {
    UFMatcher.GetErrorString(ufm_res, out m_strError);
    tbxMessage.AppendText("UFMatcher Verify: " + m_strError + "\r\n");
    return;
}
if(VerifySucceed) {
    tbxMessage.AppendText("Verification succeed (Serial = " + Serial + ")\r\n");
} else {
    tbxMessage.AppendText("Verification failed\r\n");
}
}
}

```

Delete All Database Data

Description

Database demo sample's 'Delete all' button is set to calling the OnDeleteAll function. This function deletes all database data by using UFD_RemoveAllData function.

Using functions

[UFD_RemoveAllData](#), [UFD_GetErrorString](#)

Source Code

VS60

```
void CUFE30_DatabaseDemoDlg::OnDeleteAll()
{
    <ufd_res takes UFDatabase function's return value>
    UFD_STATUS ufd_res;
    <deletes all database data>
    ufd_res = UFD_RemoveAllData(m_hDatabase);
    if(ufd_res == UFD_OK) {
        AddMessage("UFD_RemoveAllData: OK\r\n");
        UpdateDatabaseList();
    } else {
        UFD_GetErrorString(ufd_res, m_strError);
        AddMessage("UFD_RemoveAllData: %s\r\n", m_strError);
    }
}
```

VB60

```
Private Sub btnDeleteAll_Click()
    <ufd_res takes UFDatabase function's return value>
    Dim ufd_res As UFD_STATUS
    <deletes all database data>
    ufd_res = UFD_RemoveAllData(m_hDatabase)
    If(ufd_res = UFD_STATUS.OK) Then
        AddMessage ("UFD_RemoveAllData: OK" & vbCrLf)
        UpdateDatabaseList
    Else
        UFD_GetErrorString ufd_res, m_strError
        AddMessage ("UFD_RemoveAllData: " & m_strError & vbCrLf)
    End If
End Sub
```

C#

```
private void btnDeleteAll_Click(object sender, EventArgs e) {
    <ufd_res takes UFDatabase function's return value>
    UFD_STATUS ufd_res;
    <deletes all database data>
    ufd_res = m_Database.RemoveAllData();
    if(ufd_res == UFD_STATUS.OK) {
        tbxMessage.AppendText("UFDatabase RemoveAllData: OK\r\n");
        UpdateDatabaseList();
    } else {
        UFDatabase.GetErrorString(ufd_res, out m_strError);
        tbxMessage.AppendText("UFDatabase RemoveAllData: " + m_strError +
            "\r\n");
    }
}
```

Delete Database Data

Description

Database demo sample's 'Delete' button is set to calling the OnSelectionDelete function. This function deletes a data which is selected from main dialog by using UFD_RemoveDataBySerial function.

Using functions

[UFD_RemoveDataBySerial](#), [UFD_GetErrorString](#)

Source Code

VS60

```
void CUFE30_DatabaseDemoDlg::OnSelectionDelete()
{
    <ufd_res takes UFDatabase function's return value>
    UFD_STATUS ufd_res;
    int nSelected;
    int nSerial;
    nSelected = m_ctlDatabaseList.GetSelectionMark();
    if(nSelected == -1) {
        AddMessage("Select data\r\n");
        return;
    } else {
        nSerial = atoi(m_ctlDatabaseList.GetItemText(nSelected,
            DATABASE_COL_SERIAL));
    }

    <deletes a database data by using serial number>
    ufd_res = UFD_RemoveDataBySerial(m_hDatabase, nSerial);
    if(ufd_res == UFD_OK) {
        AddMessage("UFD_RemoveDataBySerial: OK\r\n");
        UpdateDatabaseList();
    } else {
        UFD_GetErrorString(ufd_res, m_strError);
        AddMessage("UFD_RemoveDataBySerial: %s\r\n", m_strError);
    }
}
```

VB60

```

Private Sub btnSelectionDelete_Click()
    <ufd_res takes UFDatabase function's return value>
    Dim ufd_res As UFD_STATUS
    Dim Serial As Long
    If (Not lvDatabaseList.SelectedItem.Selected) Then
        AddMessage ("Select data" & vbCrLf)
        Exit Sub
    Else
        Serial = Val(lvDatabaseList.SelectedItem.Text)
    End If

    <deletes a database data by using serial number>
    ufd_res = UFD_RemoveDataBySerial(m_hDatabase, Serial)
    If (ufd_res = UFD_STATUS.OK) Then
        AddMessage ("UFD_RemoveDataBySerial: OK" & vbCrLf)
        UpdateDatabaseList
    Else
        UFD_GetErrorString ufd_res, m_strError
        AddMessage ("UFD_RemoveDataBySerial: " & m_strError & vbCrLf)
    End If
End Sub

```

C#

```

private void btnSelectionDelete_Click(object sender, EventArgs e) {
    <ufd_res takes UFDatabase function's return value>
    UFD_STATUS ufd_res;
    int Serial;
    if (lvDatabaseList.SelectedItems.Count == 0) {
        tbxMessage.AppendText("Select data\r\n");
        return;
    } else {
        Serial =
            Convert.ToInt32(lvDatabaseList.SelectedItems[0].SubItems[DATABASE_COL_SERIAL].Text);
    }
    <deletes a database data by using serial number>
    ufd_res = m_Database.RemoveDataBySerial(Serial);
    if (ufd_res == UFD_STATUS.OK) {
        tbxMessage.AppendText("UFDatabase RemoveDataBySerial: OK\r\n");
        UpdateDatabaseList();
    } else {
        UFDatabase.GetErrorString(ufd_res, out m_strError);
        tbxMessage.AppendText("UFDatabase RemoveDataBySerial: " + m_strError +
            "\r\n");
    }
}

```

Update User Information

Description

Selects a data entry which you want to change about user information, from main dialog and clicks the 'Update user info' button. Then this action is set to starting the OnSelectionUpdateUserinfo function. This function updates the user data like user id, finger index, memo data by calling UFD_UpdateDataBySerial function.

Using functions

[UFD_UpdateDataBySerial](#), [UFD_GetErrorString](#)

Source Code

VS60

```
void CUFE30_DatabaseDemoDlg::OnSelectionUpdateUserinfo()
{
    CUserInfoDlg dlg;
    <ufd_res takes UFDdatabase function's return value>
    UFD_STATUS ufd_res;
    int nSelected;
    int nSerial;
    nSelected = m_ctlDatabaseList.GetSelectionMark();
    if(nSelected == -1) {
        AddMessage("Select data\r\n");
        return;
    } else {
        nSerial = atoi(m_ctlDatabaseList.GetItemText(nSelected,
            DATABASE_COL_SERIAL));
        dlg.m_strUserID = m_ctlDatabaseList.GetItemText(nSelected,
            DATABASE_COL_USERID);
        dlg.m_nFingerIndex = atoi(m_ctlDatabaseList.GetItemText(nSelected,
            DATABASE_COL_FINGERINDEX));
        dlg.m_strMemo = m_ctlDatabaseList.GetItemText(nSelected,
            DATABASE_COL_MEMO);
    }
    AddMessage("Update user data\r\n");
    AddMessage("UserID and FingerIndex will not be updated\r\n");
    if(dlg.DoModal() != IDOK) {
        AddMessage("User data input is cancelled by user\r\n");
        return;
    }
    <updates database data by using serial number>
    ufd_res = UFD_UpdateDataBySerial(m_hDatabase, nSerial, NULL, 0, NULL, 0,
```

```
dlg.m_strMemo);
if(ufd_res == UFD_OK) {
    AddMessage("UFD_UpdateDataBySerial: OK\r\n");
    UpdateDatabaseList();
} else {
    <gets an error message string according to ufd_res which is a return value of
    UFD functions>
    UFD_GetErrorString(ufd_res, m_strError);
    AddMessage("UFD_UpdateDataBySerial: %s\r\n", m_strError);
}
```

VB60

```

Private Sub btnSelectionUpdateUserInfo_Click()
    <ufd_res takes UFDDatabase function's return value>
    Dim ufd_res As UFD_STATUS
    Dim Serial As Long

    If (Not lvDatabaseList.SelectedItem.Selected) Then
        AddMessage ("Select data" & vbCrLf)
        Exit Sub
    Else
        Serial = Val(lvDatabaseList.SelectedItem.Text)
        UserInfoForm.SetUpdate
        UserInfoForm.txtUserID =
        lvDatabaseList.SelectedItem.ListSubItems(DATABASE_COL_USERID).Text
        UserInfoForm.txtFingerIndex =
        lvDatabaseList.SelectedItem.ListSubItems(DATABASE_COL_FINGERINDEX).Text
        UserInfoForm.txtMemo =
        lvDatabaseList.SelectedItem.ListSubItems(DATABASE_COL_MEMO).Text
    End If

    AddMessage ("Update user data" & vbCrLf)
    AddMessage ("UserID and FingerIndex will not be updated" & vbCrLf)
    UserInfoForm.Show vbModal
    If (Not UserInfoForm.DialogResult) Then
        AddMessage ("User data input is cancelled by user" & vbCrLf)
        Exit Sub
    End If

    <updates database data by using serial number>
    ufd_res = UFD_UpdateDataBySerial(m_hDatabase, Serial, vbNull, 0, vbNull, 0,
    UserInfoForm.txtMemo)
    If (ufd_res = UFD_STATUS.OK) Then
        AddMessage ("UFD_UpdateDataBySerial: OK" & vbCrLf)
        UpdateDatabaseList
    Else
        <gets an error message string according to ufd_res which is a return value of
        UFD functions>
    End If

```

```

    UFD_GetErrorString ufd_res, m_strError
    AddMessage ("UFD_UpdateDataBySerial: " & m_strError & vbCrLf)
End If
End Sub

```

C#

```

private void btnSelectionUpdateUserInfo_Click(object sender, EventArgs e) {
    UserInfoForm dlg = new UserInfoForm(true);
    <ufd_res takes UFDatabase function's return value>
    UFD_STATUS ufd_res;
    int Serial;
    if (lvDatabaseList.SelectedIndices.Count == 0) {
        tbxMessage.AppendText("Select data\r\n");
        return;
    } else {
        Serial =
            Convert.ToInt32(lvDatabaseList.SelectedItems[0].SubItems[DATABASE_CO
            L_SERIAL].Text);
        dlg.UserID =
            lvDatabaseList.SelectedItems[0].SubItems[DATABASE_COL_USERID].Text;
        dlg.FingerIndex =
            Convert.ToInt32(lvDatabaseList.SelectedItems[0].SubItems[DATABASE_CO
            L_FINGERINDEX].Text);
        dlg.Memo =
            lvDatabaseList.SelectedItems[0].SubItems[DATABASE_COL_MEMO].Text;
    }
    tbxMessage.AppendText("Update user data\r\n");
    tbxMessage.AppendText("UserID and FingerIndex will not be updated\r\n");
    if (dlg.ShowDialog(this) != DialogResult.OK) {
        tbxMessage.AppendText("User data input is cancelled by user\r\n");
        return;
    }
    <updates database data by using serial number>
    ufd_res = m_Database.UpdateDataBySerial(Serial, null, 0, null, 0, dlg.Memo);
    if (ufd_res == UFD_STATUS.OK) {
        tbxMessage.AppendText("UFD_UpdateDataBySerial: OK\r\n");
        UpdateDatabaseList();
    } else {
        <gets an error message string according to ufd_res which is a return value of
        UFD functions>
        UFDatabase.GetErrorString(ufd_res, out m_strError);
        tbxMessage.AppendText("UFDatabase UpdateDataBySerial: " + m_strError +
        "\r\n");
    }
}

```

Update User Template

Description

Selects a data entry which you want to change the template, from the database dialog and clicks the 'Update template' button. Then this action is set to calling the OnSelectionUpdateTemplate function. This function extracts a template from user's finger by using ExtractTemplate(VS60) function. And then updates the template data by calling UFD_UpdateDataBySerial function.

Using functions

[UFD_UpdateDataBySerial](#), [UFS_ClearCaptureImageBuffer](#) in ExtractTemplate,
[UFS_CaptureSingleImage](#) in ExtractTemplate, [UFS_Extract](#) in ExtractTemplate,
[UFD_GetErrorString](#)

Source Code

VS60

```
void CUFE30_DatabaseDemoDlg::OnSelectionUpdateTemplate()
{
    <ufd_res takes UFDatabase function's return value>
    UFD_STATUS ufd_res;
    int nSelected;
    int nSerial;
    nSelected = m_ctlDatabaseList.GetSelectionMark();
    if(nSelected == -1) {
        AddMessage("Select data\r\n");
        return;
    } else {
        nSerial = atoi(m_ctlDatabaseList.GetItemText(nSelected,
            DATABASE_COL_SERIAL));
    }
    <extracts a template after capturing a fingerprint. The extracted template data will be
    saved in the Template variable>
    if(!ExtractTemplate(m_Template1, &m_nTemplate1Size)) {
        return;
    }
    <updates database data by using serial number>
    ufd_res = UFD_UpdateDataBySerial(m_hDatabase, nSerial, m_Template1,
        m_nTemplate1Size, NULL, 0, NULL);
    if(ufd_res == UFD_OK) {
        AddMessage("UFD_UpdateDataBySerial: OK\r\n");
        UpdateDatabaseList();
    } else {
        <gets an error message string according to ufd_res which is a return value of
        UFD_GetErrorString>
    }
}
```

```
UFD functions>
UFD_GetErrorString(ufd_res, m_strError);
AddMessage("UFD_UpdateDataBySerial: %s\r\n", m_strError);
```

VB60

```

Private Sub btnSelectionUpdateTemplate_Click()
    <ufd_res takes UFDDatabase function's return value>
    Dim ufd_res As UFD_STATUS
    Dim Serial As Long

    If (Not lvDatabaseList.SelectedItem.Selected) Then
        AddMessage ("Select data" & vbCrLf)
        Exit Sub
    Else
        Serial = Val(lvDatabaseList.SelectedItem.text)
    End If

    <extracts a template after capturing a fingerprint. The extracted template data will be
    saved in the Template variable>
    If (Not ExtractTemplate(m_Template1, m_Template1Size)) Then
        Exit Sub
    Else
        pblImageFrame.Refresh
    End If

    <updates database data by using serial number>
    ufd_res = UFD_UpdateDataBySerial(m_hDatabase, Serial, m_Template1(0),
    m_Template1Size, vbNull, 0, vbNullString)
    If (ufd_res = UFD_STATUS.OK) Then
        AddMessage ("UFD_UpdateDataBySerial: OK" & vbCrLf)
        UpdateDatabaseList
    Else
        <gets an error message string according to ufd_res which is a return value of
        UFD functions>
        UFD_GetErrorString ufd_res, m_strError
        AddMessage ("UFD_UpdateDataBySerial: " & m_strError & vbCrLf)
    End If
End Sub

```

C#

```
private void btnSelectionUpdateTemplate_Click(object sender, EventArgs e) {
    <ufd_res takes UFDATABASE function's return value>
    UFD_STATUS ufd_res;
    int Serial;
    if (lvDatabaseList.SelectedIndices.Count == 0) {
        tbxMessage.AppendText("Select data\r\n");
    }
}
```

```
        return;
    } else {
        Serial =
            Convert.ToInt32(lvDatabaseList.SelectedItems[0].SubItems[DATABASE_CO
L_SERIAL].Text);
    }
    <extracts a template after capturing a fingerprint. The extracted template data will be
    saved in the Template variable>
    if (!ExtractTemplate(m_Template1, out m_Template1Size)) {
        return;
    }
    DrawCapturedImage(m_Scanner);
    <updates database data by using serial number>
    ufd_res = m_Database.UpdateDataBySerial(Serial, m_Template1, m_Template1Size,
        null, 0, null);
    if (ufd_res == UFD_STATUS.OK) {
        tbxMessage.AppendText("UFD_UpdateDataBySerial: OK\r\n");
        UpdateDatabaseList();
    } else {
        <gets an error message string according to ufd_res which is a return value of
        UFD functions>
        UFDatabase.GetErrorString(ufd_res, out m_strError);
        tbxMessage.AppendText("UFDatabase UpdateDataBySerial: " + m_strError +
            "\r\n");
    }
}
```

UFE30_EnrollDemo

UFE30_EnrollDemo provides the basic usage about managing scanners and executing enrollment. This program uses [UFScanner](#) module.

Required products

Products	Remarks
Suprema Biomini Enroll SDK	No restriction

Available languages

Languages	Locations
Visual C++ 6.0	samples\VS60\UFE30_EnrollDemoVC60
Visual Basic 6.0	samples\VS60\UFE30_EnrollDemoVB60
Visual C#	samples\VS80\UFE30_EnrollDemoCS
Visual Basic .NET	samples\VS80\UFE30_EnrollDemoVBNET

UFE30_EnrollDemo_Usage

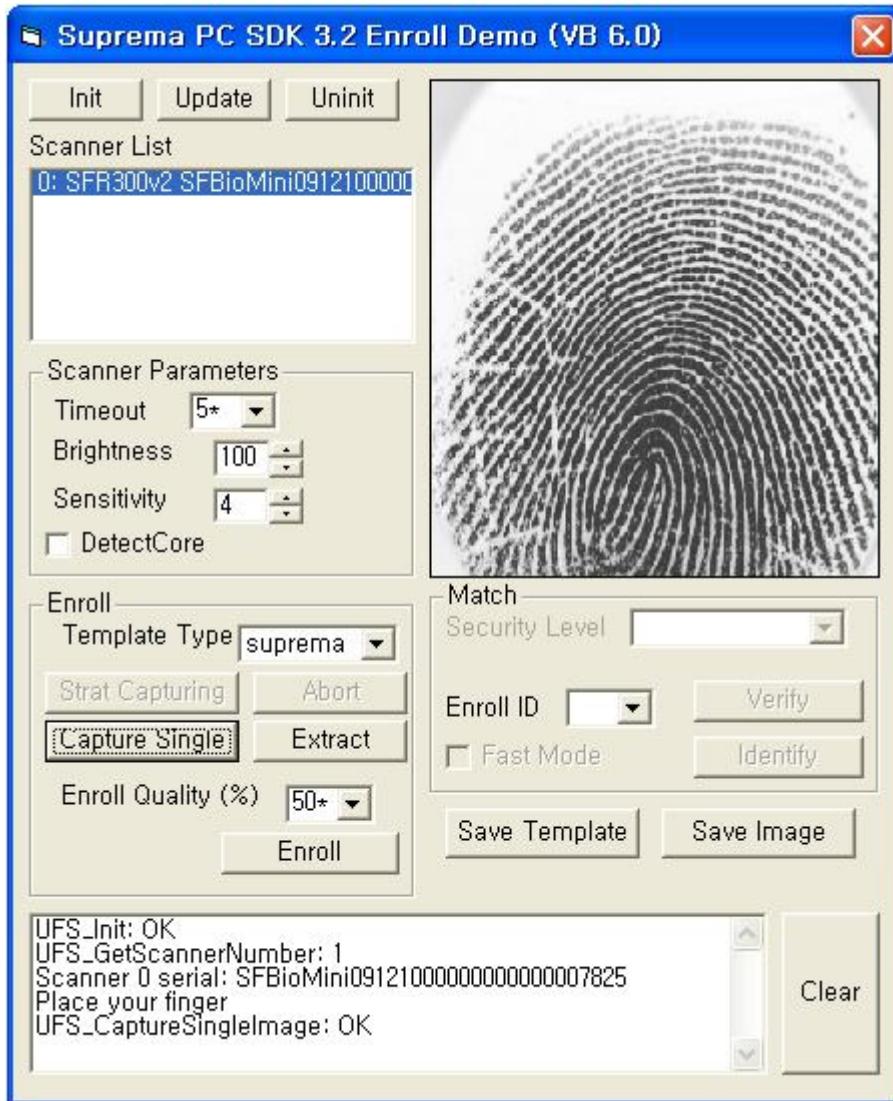
Executable File Location

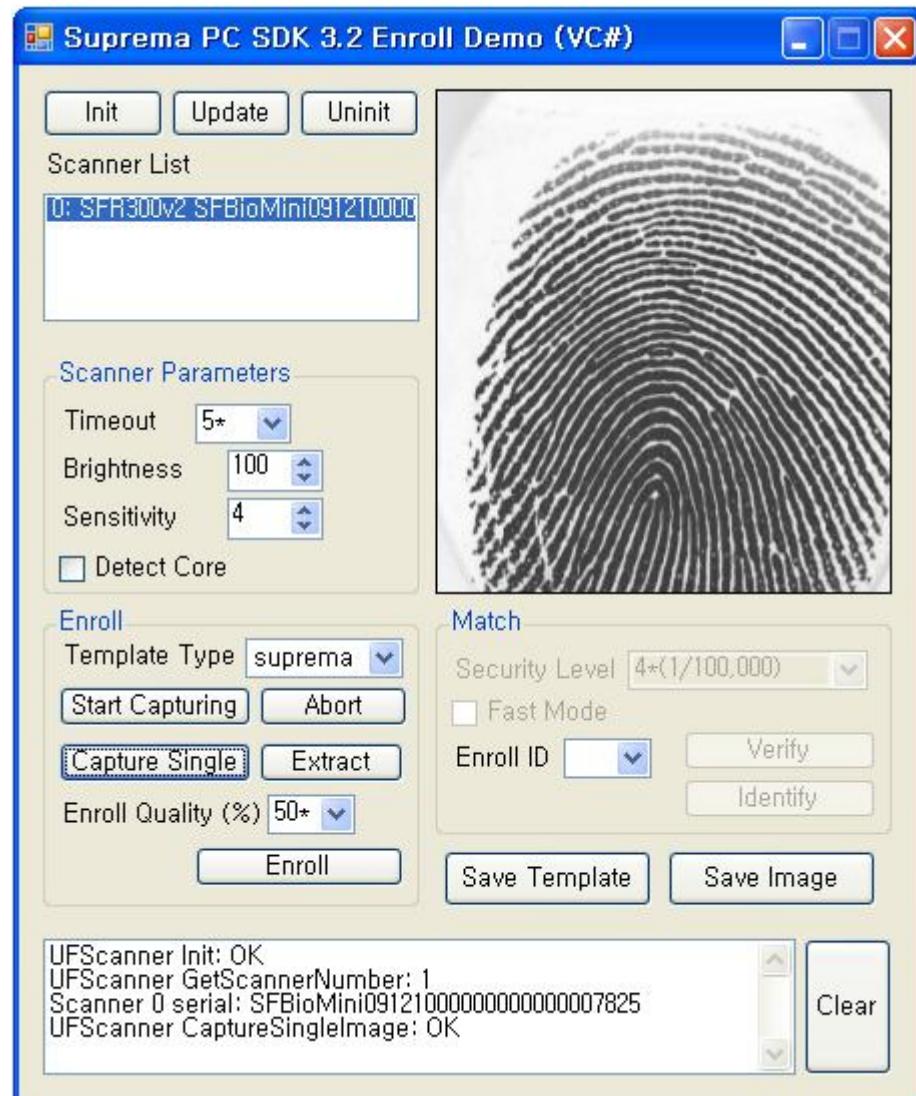
- bin\UFE30_EnrollDemoCS.exe
- bin\UFE30_EnrollDemoVBNET.exe
- bin\UFE30_EnrollDemoVC60.exe
- Source code of all demo application

Required products

[Suprema Biomini Enroll SDK](#)

Picture of the Demo sample applications





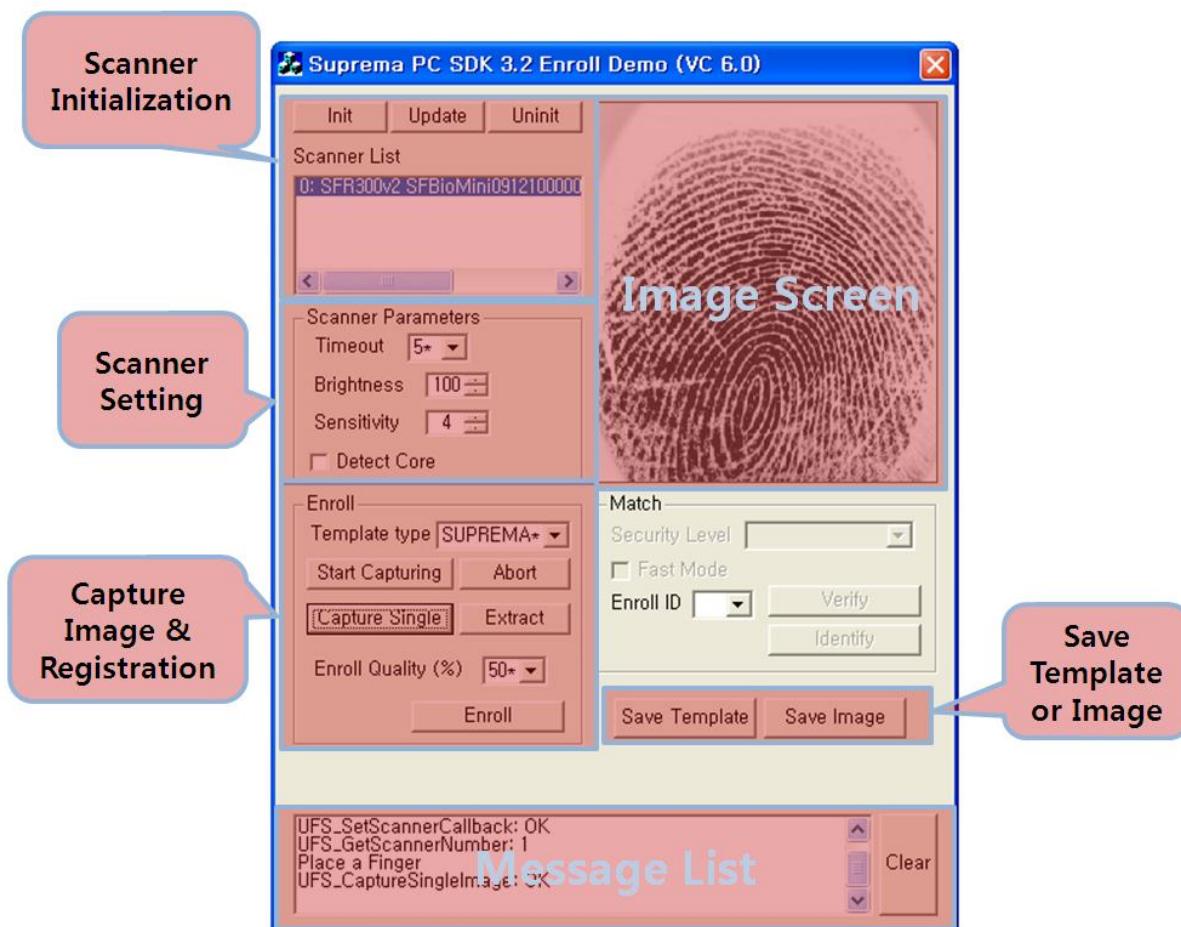
User Interface Components

Suprema PC SDK EnrollDemo sample demonstrates how to use Suprema PC SDK about template extraction and enroll process . Please use this sample as a reference for making your own program. PC SDK and Enroll SDK, Enroll demo provides following methods:

- Enroll fingerprint - A fingerprint can be enrolled by using BioMini scanner
- Extract fingerprint template - Extracts a template from captured fingerprint image

Every UFE30_EnrollDemo application is written by different languages from Visual C++ to C#. But they are made of same functions and same interface design.

The picture bellow shows the PC SDK EnrollDemo sample main window and description about bunch of interface components :



The table bellow shows the PC SDK EnrollDemo's simple functionality and links to each detailed usage page :

Scanner initialize
The items about scanner initialization. Using these components, to makes a connection or disconnection with fingerprint scanner and checking the connection state

Init button - Initializes UFScanner module	Source Code
Update button - Checks the condition of connection state with scanner	Source Code
Uninit button - Uninitializes UFScanner module	Source Code
ScannerList list box - List of the scanners	

Scanner options
Option items for the capturing process
Timeout combo box - Countdown value for the 'capture' functions
Brightness spin Edit - Sets the brightness of connected scanner
Sensitivity spin Edit - Sets the sensitivity of connected scanner
Detect Core check box - If this check box is checked the scanner captures an image only when find the 'core' of fingerprint
Source Code

Enroll
To capture a fingerprint image and extracts a template from captured image
Template type combo box - Sets the template type with extraction function
Start Capturing button - Captures an image with callback function
Abort button - Cancels start capturing
Source Code
Caputre Single button - Captures a fingerprint image
Source Code
Extract button - Extracts a template from captured image
Source Code
Enroll Quality (%) Combo box - the Quality threshold
Enroll button - Enrolls a fingerprint data
Source Code

Save Template - Saves an image Template
Save Image - Saves an image of bmp format

UFE30_MultiScannerDemo

UFE30_MultiScannerDemo provides the demo about using multiple scanners simultaneously. This program uses [UFScanner](#) and [UFMatcher](#) modules.

Required products

Products	Remarks
Suprema BioMini SDK	No restriction

Available languages

Languages	Locations
Visual C++ 6.0	samples\VS60\UFE30_MuliScannerDemoVC60

UFE_MultiScannerDemo

UFE30_MultiScanner_Usage

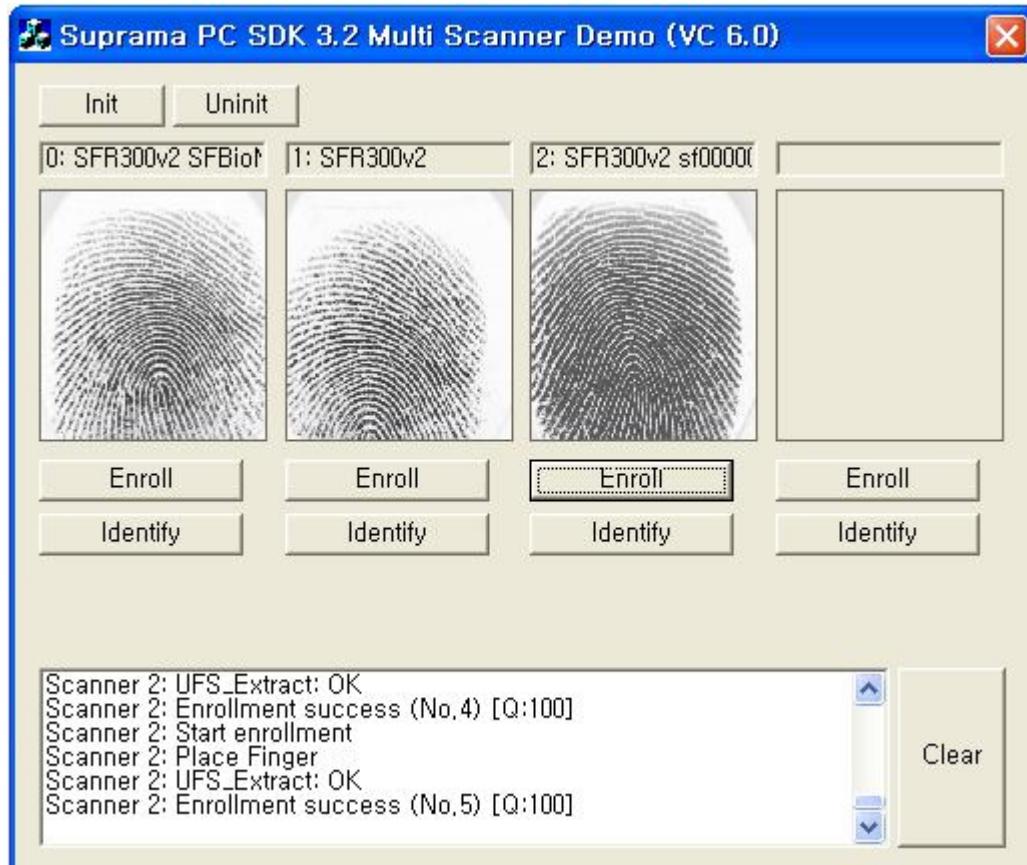
Executable File Location

- bin\UFE30_MultiScannerDemoVC60.exe
- Source code of all demo application

Required products

[Suprema BioMini SDK](#)

Picture of the Demo sample applications



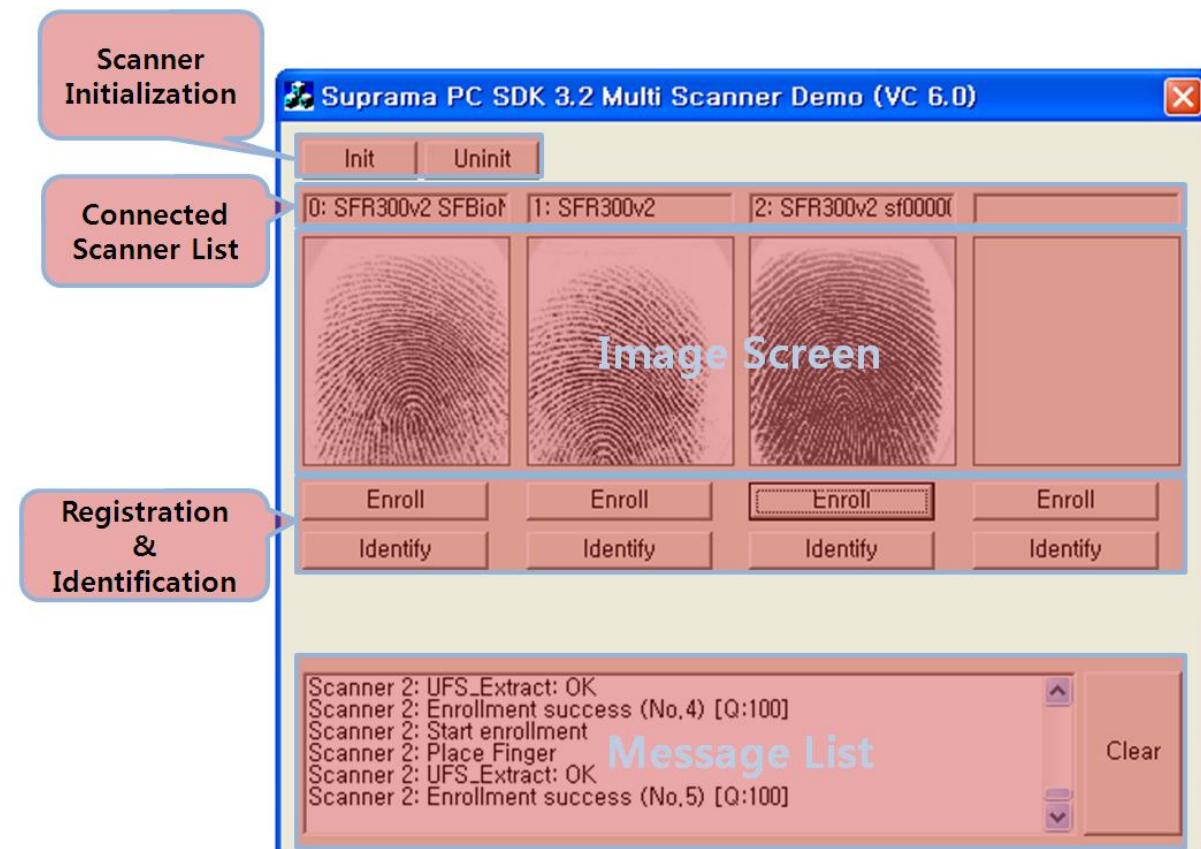
User Interface Components

Suprema PC SDK MultiScannerDemo sample demonstrates how to use Suprema PC SDK with multiple scanners. Please use this sample as a reference for making your own program. PC SDK Multiscanner demo provides following methods:

- Initialize Scanner - The scanners should be initialized for using the functions about the scanner
- Enroll fingerprint - A fingerprint can be enrolled by using BioMini scanner
- Identification - A fingerprint can be identified against every enrolled fingerprint

UFE30_MultiScannerDemo application written in only Visual C++ language(VS6.0).

The picture bellow shows the PC SDK MultiScannerDemo sample main window and description about bunch of interface components :



The table bellow shows the PC SDK MultiScannerDemo's simple functionality and links to each detailed usage page :

Scanner initialize
The items about scanner initialization. Using these components, to makes a connection or disconnection with the scanner and checking the connection state

Init button - Initializes UFScanner module Code	Source
Uninit button - Uninitializes UFScanner module Source Code	

Enroll and Identify

The MultiScanner sample includes two fingerprint processing	
Enroll button - Enrolls a fingerprint data Code	Source
Identify button - Identifies input fingerprint from enrolled fingerprints Code	Source

MultiScanner Initialization

Description

To capture a fingerprint image, one or more scanner has to be initialized with your program. This page is the source code about initialization task that is connected with 'multiscanner' button in 'database' samples.

Using functions

[UFS_Init](#), [UFS_GetErrorString](#), [UFS_GetScannerNumber](#), [UFS_GetScannerHandle](#),
[UFS_GetScannerID](#), [UFS_GetScannerType](#), [UFM_Create](#), [UFM_GetErrorString](#)

wrapper

Source Code

OnInit	
<pre>void CUFE30_MultiScannerDemoDlg::OnInit() { ///////////////// // Initialize scanners and matchers /////////////// <ufs_res takes UFSanner function's return value> UFS_STATUS ufs_res; <ufm_res takes UFMatcher function's return value> UFM_STATUS ufm_res; <the number of connected scanners> int nScannerNumber; int i; BeginWaitCursor(); <scanner initialization function. see UFS_Init to check the return values. This function returns true value if the initialization success false otherwise> ufs_res = UFS_Init(); EndWaitCursor(); if(ufs_res == UFS_OK) { AddMessage("UFS_Init: OK\r\n"); } else { <gets an error message string according to ufs_res which is a return value of UFS functions> UFS_GetErrorString(ufs_res, m_strError); AddMessage("UFS_Init: %s\r\n", m_strError); } }</pre>	

```

        return;
    }
    <gets the number of connected scanners>
    ufs_res = UFS_GetScannerNumber(&nScannerNumber);
    if(ufs_res == UFS_OK) {
        AddMessage("UFS_GetScannerNumber: %d\r\n", nScannerNumber);
    } else {
        UFS_GetErrorString(ufs_res, m_strError);
        AddMessage("UFS_GetScannerNumber: %s\r\n", m_strError);
    }
    if(nScannerNumber == 0) {
        return;
    }
    for (i = 0; i < nScannerNumber; i++) {
        // Only first 4 scanners will be used
        if(i >= 4) {
            break;
        }
        int nScannerType;
        char strScannerType[64];
        char strID[64];
        char strTmp[256];
        <gets a scanner handle by index>
        UFS_GetScannerHandle(i, &m_hScanner[i]);
        <gets the scanner type>
        UFS_GetScannerType(m_hScanner[i], &nScannerType);
        <gets the serial number>
        UFS_GetScannerID(m_hScanner[i], strID);
        GetScannerTypeString(nScannerType, strScannerType);
        sprintf(strTmp, "%d: %s %s", i, strScannerType, strID);
        m_strScannerID[i].Format("%d: %s %s", i, strScannerType, strID);
        <creates a matcher which is responsible for the matching process>
        ufm_res = UFM_Create(&m_hMatcher[i]);
        if(ufm_res == UFM_OK) {
            AddMessage("UFM_Create: OK\r\n");
        } else {
            <gets an error message string according to ufm_res which is a return
            value of UFM function>
            UFM_GetErrorString(ufm_res, m_strError);
            AddMessage("UFM_Create: %s\r\n", m_strError);
            return;
        }
        m_bIsScannerBusy[i] = FALSE;
    }
    UpdateData(FALSE);
    Invalidate(FALSE);
    /////////////////////////////////
}

```

MultiScanner Uninitialization

Description

The multiscanner sample's 'uninit' button is set to calling the OnUninit(VS60) function. To use the scanner properly and prevent unpredictable errors, scanner resources should be released when they are unused. Use the UFS_Uninit function to release the resources that were allocated at initialization.

Using functions

[UFS_Uninit](#), [UFS_GetErrorString](#), [UFM_Delete](#), [UFM_GetErrorString](#)

Source Code

OnUninit
<pre>void CUFE30_MultiScannerDemoDlg::OnUninit() { // // Uninit scanners // <ufs_res takes UFSanner function's return value> UFS_STATUS ufs_res; BeginWaitCursor(); <uninitializes a scanner, use this to release resources which are related to the scanner. See UFS_Uninit to check the return value> ufs_res = UFS_Uninit(); EndWaitCursor(); if(ufs_res == UFS_OK) { AddMessage("UFS_Uninit: OK\r\n"); m_strScannerID[0] = ""; m_strScannerID[1] = ""; m_strScannerID[2] = ""; m_strScannerID[3] = ""; UpdateData(FALSE); Invalidate(TRUE); } else { <gets an error message string according to ufs_res which is a return value of UFS function> UFS_GetErrorString(ufs_res, m_strError); AddMessage("UFS_Uninit: %s\r\n", m_strError); } // // Delete matchers //</pre>

```
UFM_STATUS ufm_res;
int i;
for (i = 0; i < 4; i++) {
    if(m_hMatcher[i] != NULL) {
        <releases matcher's resources. See UFM\_Delete>
        ufm_res = UFM_Delete(m_hMatcher[i]);
        if(ufm_res == UFM_OK) {
            AddMessage("UFM_Delete: OK\r\n");
        } else {
            <gets an error message string according to ufm_res which is a
            return value of UFM function>
            UFM_GetErrorString(ufm_res, m_strError);
            AddMessage("UFM_Delete: %s\r\n", m_strError);
        }
        m_hMatcher[i] = NULL;
    }
}
///////////////////////////////////////////////////////////////////
```

MultiScanner Enrollment

Description

In the MultiScanner sample, each 'Enroll' button connect with each OnEnroll[Number] function. This function starts a thread by calling StartEnrollThread function. This function uses a scanner index number to classify each thread. To enroll a fingerprint, the thread captures a fingerprint image and calls the UFS_Extract function to extract a template from the captured image. This template will be saved in a specific array which is a parameter of the UFS_Extract function.

Using functions

[UFS_CaptureSingleImage](#), [UFS_Extract](#), [UFS_GetErrorString](#),
[UFS_DrawCaptureImageBuffer](#)

Source Code

OnInit	
<pre>////////// // Enroll Thread Functions ////////// typedef struct { int nScannerIdx; CUFE30_MultiScannerDemoDlg* pDlg; } EnrollParam; UINT EnrollThread(LPVOID pParam) { EnrollParam* pEnrollParam = (EnrollParam*)pParam; <gets the scanner index> int nScannerIdx = pEnrollParam->nScannerIdx; CUFE30_MultiScannerDemoDlg* pDlg = pEnrollParam->pDlg; <ufs_res takes UFSanner function's return value> UFS_STATUS ufs_res; int nEnrollQuality; int i; pDlg->AddMessage("Scanner %d: Place Finger\r\n", nScannerIdx); for (i = 0;; i++) { <scanner is standby for capturing> ufs_res = UFS_CaptureSingleImage(pDlg->m_hScanner[nScannerIdx]); if (ufs_res != UFS_OK) { <gets an error message string according to ufs_res which is a return value of UFS function> UFS_GetErrorString(ufs_res, pDlg->m_strError); pDlg->AddMessage("Scanner %d: UFS_CaptureSingleImage: %s\r\n",</pre>	

```

        nScannerIdx, pDlg->m_strError);
        goto errret;
    }
    <extracts a template from captured fingerprint image and saves the enrolled
     template in pDlg->m_template[nScannerIdx][pDlg-
     >m_template_num[nScannerIdx]] array>
    ufs_res = UFS_Extract(pDlg->m_hScanner[nScannerIdx], pDlg-
    >m_template[nScannerIdx][pDlg->m_template_num[nScannerIdx]], &pDlg-
    >m_template_size[nScannerIdx][pDlg->m_template_num[nScannerIdx]],
    &nEnrollQuality);
    if(ufs_res == UFS_OK) {
        pDlg->AddMessage("Scanner %d: UFS_Extract: OK\r\n", nScannerIdx);
        pDlg->Invalidate(FALSE);
        CRect rect;
        pDlg->m_ctlImageFrame[nScannerIdx].GetClientRect(&rect);
        rect.DeflateRect(1, 1);
        <draws a captured image at DC>
        UFS_DrawCaptureImageBuffer(pDlg->m_hScanner[nScannerIdx],
        pDlg->m_ctlImageFrame[nScannerIdx].GetDC()->m_hDC, rect.left,
        rect.top, rect.right, rect.bottom, TRUE);
        break;
    } else {
        UFS_GetErrorString(ufs_res, pDlg->m_strError);
        pDlg->AddMessage("Scanner %d: UFS_Extract: %s\r\n", nScannerIdx,
        pDlg->m_strError);
    }
}
if(nEnrollQuality < 2 * 10 + 30 ) {
    pDlg->AddMessage("Scanner %d: Too low quality [Q:%d]\r\n",
    nScannerIdx, nEnrollQuality);
    goto errret;
}
pDlg->AddMessage("Scanner %d: Enrollment success (No.%d) [Q:%d]\r\n",
nScannerIdx, pDlg->m_template_num[nScannerIdx]+1, nEnrollQuality);
if(pDlg->m_template_num[nScannerIdx]+1 == MAX_TEMPLATE_NUM) {
    pDlg->AddMessage("Scanner %d: Template memory is full\r\n",
    nScannerIdx);
} else {
    pDlg->m_template_num[nScannerIdx]++;
}
errret:
free(pEnrollParam);
pDlg->m_bIsScannerBusy[nScannerIdx] = FALSE;
return 1;
}

<sets the basic resources of enrollthread for No.[nScannerIdx] scanner and starts the
thread>
void CUFE30_MultiScannerDemoDlg::StartEnrollThread(int nScannerIdx)

```

```
{  
    if(m_bIsScannerBusy[nScannerIdx]) {  
        AddMessage("Scanner %d: Scanner is busy\r\n", nScannerIdx);  
        return;  
    } else {  
        AddMessage("Scanner %d: Start enrollment\r\n", nScannerIdx);  
    }  
    m_bIsScannerBusy[nScannerIdx] = TRUE;  
    EnrollParam* pEnrollParam;  
    pEnrollParam = (EnrollParam*)malloc(sizeof(EnrollParam));  
    pEnrollParam->nScannerIdx = nScannerIdx;  
    pEnrollParam->pDlg = this;  
    AfxBeginThread(EnrollThread, (LPVOID)pEnrollParam);  
}  
  
<starts enrollment>  
void CUFE30_MultiScannerDemoDlg::OnEnroll1()  
{  
    StartEnrollThread(0);  
}  
void CUFE30_MultiScannerDemoDlg::OnEnroll2()  
{  
    StartEnrollThread(1);  
}  
void CUFE30_MultiScannerDemoDlg::OnEnroll3()  
{  
    StartEnrollThread(2);  
}  
void CUFE30_MultiScannerDemoDlg::OnEnroll4()  
{  
    StartEnrollThread(3);  
}
```

MultiScanner Identification

Description

In the MultiScanner sample, each 'Identify' button connects with each OnIdentify[Number] function. This function starts a thread by calling StartIdentifyThread function, using the scanner index as parameter value. To identify a fingerprint, the thread captures a fingerprint image and extracts a template from the image. And then calls the UFM_Identify function to execute 1:N matching using the extracted template and all the other saved templates.

Using functions

[UFS_ClearCaptureImageBuffer](#), [UFS_GetErrorString](#), [UFS_Extract](#),
[UFS_DrawCaptureImageBuffer](#), [UFM_Identify](#), [UFM_GetErrorString](#)

Source Code

OnIdentify

```
//////////  

// Identify Thread Functions  

//////////  

typedef struct {  

    int nScannerIdx;  

    CUFE30_MultiScannerDemoDlg* pDlg;  

} IdentifyParam;  

UINT IdentifyThread(LPVOID pParam) {  

    IdentifyParam* pIdentifyParam = (IdentifyParam*)pParam;  

    int nScannerIdx = pIdentifyParam->nScannerIdx;  

    CUFE30_MultiScannerDemoDlg* pDlg = pIdentifyParam->pDlg;  

    <ufs_res takes UFSanner function's return value>  

    UFS_STATUS ufs_res;  

    <ufm_res takes UFMatcher function's return value>  

    UFM_STATUS ufm_res;  

    unsigned char Template[MAX_TEMPLATE_SIZE];  

    int TemplateSize;  

    int nEnrollQuality;  

    int nMatchIndex;  

    int i;  

    pDlg->AddMessage("Scanner %d: Place Finger\r\n", nScannerIdx);  

    UFS_ClearCaptureImageBuffer(pDlg->m_hScanner[nScannerIdx]);  

    for (i = 0;; i++) {  

        <scanner is standby for capturing>  

        ufs_res = UFS_CaptureSingleImage(pDlg->m_hScanner[nScannerIdx]);  

        if(ufs_res != UFS_OK) {  

            <gets an error message string according to ufs_res which is a return value>
        }
    }
}
```

```

    of UFS function>
    UFS_GetErrorString(ufs_res, pDlg->m_strError);
    pDlg->AddMessage("Scanner %d: UFS_CaptureSingleImage: %s\r\n",
    nScannerIdx, pDlg->m_strError);
    goto errret;
}
<extracts a template from captured fingerprint image and saves the template in
template variable>
ufs_res = UFS_Extract(pDlg->m_hScanner[nScannerIdx], Template,
&TemplateSize, &nEnrollQuality);
if(ufs_res == UFS_OK) {
    pDlg->AddMessage("Scanner %d: UFS_Extract: OK\r\n", nScannerIdx);
    pDlg->Invalidate(FALSE);
    CRect rect;
    pDlg->m_ctlImageFrame[nScannerIdx].GetClientRect(&rect);
    rect.DeflateRect(1, 1);
    <draws captured image at the DC>
    UFS_DrawCaptureImageBuffer(pDlg->m_hScanner[nScannerIdx],
    pDlg->m_ctlImageFrame[nScannerIdx].GetDC()->m_hDC, rect.left,
    rect.top, rect.right, rect.bottom, TRUE);
    break;
} else {
    UFS_GetErrorString(ufs_res, pDlg->m_strError);
    pDlg->AddMessage("Scanner %d: UFS_Extract: %s\r\n", nScannerIdx,
    pDlg->m_strError);
}
}
<identifies a fingerprint template against all saved fingerprint templates>
ufm_res = UFM_Identify(pDlg->m_hMatcher[nScannerIdx], Template,
TemplateSize, pDlg->m_template[nScannerIdx], pDlg-
>m_template_size[nScannerIdx], pDlg->m_template_num[nScannerIdx], 5000,
&nMatchIndex);
if(ufm_res != UFM_OK) {
    <gets an error message string according to ufm_res which is a return value of
    UFM function>
    UFM_GetErrorString(ufm_res, pDlg->m_strError);
    pDlg->AddMessage("Scanner %d: UFM_Identify: %s\r\n", nScannerIdx,
    pDlg->m_strError);
    goto errret;
}
if(nMatchIndex != -1) {
    pDlg->AddMessage("Scanner %d: Identification succeed (No.%d)\r\n",
    nScannerIdx, nMatchIndex+1);
} else {
    pDlg->AddMessage("Scanner %d: Identification failed\r\n", nScannerIdx);
}
errret:
free(pIdentifyParam);
pDlg->m_bIsScannerBusy[nScannerIdx] = FALSE;

```

```

        return 1;
    }

<sets the basic resources of identifythread for No.[nScannerIdx] scanner and starts the
thread>
void CUFE30_MultiScannerDemoDlg::StartIdentifyThread(int nScannerIdx)
{
    if(m_bIsScannerBusy[nScannerIdx]) {
        AddMessage("Scanner %d: Scanner is busy\r\n", nScannerIdx);
        return;
    } else {
        AddMessage("Scanner %d: Start identification\r\n", nScannerIdx);
    }
    m_bIsScannerBusy[nScannerIdx] = TRUE;
    IdentifyParam* pIdentifyParam;
    pIdentifyParam = (IdentifyParam*)malloc(sizeof(IdentifyParam));
    pIdentifyParam->nScannerIdx = nScannerIdx;
    pIdentifyParam->pDlg = this;
    AfxBeginThread(IdentifyThread, (LPVOID)pIdentifyParam);
}

<starts identification>
void CUFE30_MultiScannerDemoDlg::OnIdentify1()
{
    StartIdentifyThread(0);
}
void CUFE30_MultiScannerDemoDlg::OnIdentify2()
{
    StartIdentifyThread(1);
}
void CUFE30_MultiScannerDemoDlg::OnIdentify3()
{
    StartIdentifyThread(2);
}
void CUFE30_MultiScannerDemoDlg::OnIdentify4()
{
    StartIdentifyThread(3);
}

```

Chapter 5. Reference

This chapter contains reference of all modules included in Suprema PC SDK for developers using following languages,

- Visual C++ 6.0 / 7.0 / 7.1 / 8.0
- Visual Basic 6.0 ¹⁾
- Java SDK 1.4 or higher

APIs are described using C language. Modules are created using Visual C++ 6.0 and they are compatible with Visual C++ 7.0 / 7.1 / 8.0 or higher.

¹⁾ For accessing C/C++ style API in Visual Basic 6.0, we supply Suprema PC SDK 3.1 Type Library for Visual Basic 6.0 (bin\Suprema.tlb). Users can add this type library to their projects using browse button in the References dialog (drop down the Project menu and select the References item). As Suprema type library is simple wrapper of Suprema PC SDK modules, we do not add distinct references for the type library. The parameter definitions of the type library can be shown using object browser and usage of the type library is described in [tutorial](#) and [sample](#).

UFScanner module

UFScanner module provides functionality for managing scanners, capturing finger images from scanners, extracting templates from captured images using scanners, etc.

Requirements

Visual C++
<ul style="list-style-type: none"> Required header: include\UFScanner.h Required lib: lib\UFScanner.lib Required dll: bin\UFScanner.dll
Visual Basic 6.0
<ul style="list-style-type: none"> Required reference: Suprema type library (bin\Suprema.tlb) Required dll: bin\UFScanner.dll

Supported scanners

Scanner	Scanner name in module	Support for multi-device
Suprema SFR200	SFR200	X
Suprema SFR300-S	SFR300	X
Suprema SFR300-S(Ver.2)	SFR300v2	O

Definitions

Status return value (UFS_STATUS)

Every function in UFScanner module(UFScannerClass method) returns UFS_STATUS (integer) value having one of following values,

Status value definition	Code	Meaning
UFS_OK	0	Success
UFS_ERROR	-1	General error
UFS_ERR_NO_LICENSE	-101	System has no license
UFS_ERR_LICENSE_NOT_MATCH	-102	License is not match
UFS_ERR_LICENSE_EXPIRED	-103	License is expired
UFS_ERR_NOT_SUPPORTED	-111	This function is not supported
UFS_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
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 is exceed
UFS_ERR_LOAD_SCANNER_LIBRARY	-204	Error on loading scanner library
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_NOT_GOOD_IMAGE	-301	Input image is not good
UFS_ERR_EXTRACTION_FAILED	-302	Extraction is failed
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

Parameters

[UFS_GetParameter\(\)](#), [UFS_SetParameter\(\)](#) functions use parameters defined as follows,

Parameter value definition	Code	Meaning	Default value
UFS_PARAM_TIMEOUT	201	Timeout (millisecond unit) (0: infinite)	5000
UFS_PARAM_BRIGHTNESS	202	Brightness (0 ~ 255); Higher value means darker image	100
UFS_PARAM_SENSITIVITY	203	Sensitivity (0 ~ 7); Higher value means more sensitive	4
UFS_PARAM_SERIAL	204	Serial (get only)	
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)	384
UFS_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0
UFS_PARAM_DETECT_FAKE	312	Use Fake Finger Detection (0: not use LFD, 1~3 : use LFD); Higher value means more strong to fake finger	0

Template type

[UFS_GetTemplateType\(\)](#), [UFS_SetTemplateType\(\)](#) functions use parameters defined as follows,

template type definition	Code	Meaning
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

Scanner type

[UFS_GetScannerType\(\)](#) function gives type values defined as follows,

Scanner type definition	Code	Meaning
UFS_SCANNER_TYPE_SFR200	1001	Suprema SFR200
UFS_SCANNER_TYPE_SFR300	1002	Suprema SFR300-S
UFS_SCANNER_TYPE_SFR300v2	1003	Suprema SFR300-S(Ver.2)

Scanner handle

HUFScanner defines handle to UFScanner object.

```
typedef void* HUFScanner;
```

*java JNA pointer type defines handle to UFScannerClass

```
public Pointer hScanner=null;
```

Scanner callback function

UFS_SCANNER_PROC defines scanner callback function.

```
typedef int UFS_CALLBACK UFS_CALLBACK UFS_SCANNER_PROC(  
    const char* szScannerID,  
    int bSensorOn,  
    void* pParam  
)
```

*java UFS_SCANNER_PROC defines scanner callback interface.

```
interface UFS_SCANNER_PROC extends Callback,StdCall  
{  
    int callback(String szScannerId,int bSeonsorOn,PointerByReference pParam);  
}
```

Return value

1	Received event is successfully processed
0	Received event is not processed

Parameters

szScannerID	ID of the scanner which is occurred this event
bSensorOn	1: scanner is connected, 0: scanner is disconnected
pParam	Receives the scanner callback data passed to the UFS_SetScannerCallback function using the pParam parameter

Capture callback function

UFS_CAPTURE_PROC defines capture callback function

```
typedef int UFS_CALLBACK UFS_CALLBACK UFS_CAPTURE_PROC(
    HUFSscanner hScanner,
    int bFingerOn,
    unsigned char* pImage,
    int nWidth,
    int nHeight,
    int nResolution,
    void* pParam
);
```

*java UFS_CAPTURE_PROC defines capture callback interface.

```
interface UFS_CAPTURE_PROC extends Callback,StdCall
{
    int callback(Pointer hScanner, int bFingerOn, Pointer pImage, int nWidth, int nHeight, int
    nResolution, PointerByReference pParam);
}
```

Return value

1	Continue capturing
0	Finish capturing

Parameters

hScanner	Handle to the scanner object
bFingerOn	1: finger is on the scanner, 0: finger is not on the scanner
pImage	Point to buffer storing received image
nWidth	Width of received image
nHeight	Height of received image
nResolution	Resolution of received image
pParam	Receives the capture callback data passed to the UFS_StartCapturing function using the pParam parameter

UFS_Init

Initializes scanner module.

```
UFS\_STATUS UFS_API UFS_Init();  
*java  
int UFS_Init();
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_ALREADY_INITIALIZED](#),
[UFS_ERR_NO_LICENSE](#), [UFS_ERR_LICENSE_NOT_MATCH](#),
[UFS_ERR_LICENSE_EXPIRED](#), [UFS_ERR_DEVICE_NUMBER_EXCEED](#)

See also

[UFS_Update](#), [UFS_Uninit](#)

Examples

Visual C++
UFS_STATUS ufs_res; ufs_res = UFS_Init(); if(ufs_res == UFS_OK) { // UFS_Init is succeeded } else { // UFS_Init is failed // Use UFS_GetErrorString function to show error string }

Visual Basic 6.0
Dim ufs_res As UFS_STATUS ufs_res = UFS_Init() If (ufs_res = UFS_STATUS.OK) Then ' UFS_Init is succeeded Else ' UFS_Init is failed ' Use UFS_GetErrorString function to show error string End If

java JNA

```
int nRes =0;
UFSscannerClass libScanner=null;

//load library
libScanner = (UFSscannerClass)Native.loadLibrary("UFSscanner",UFSscannerClass.class);

nRes = libScanner.UFS_Init();

if(nRes ==libScanner.UFS_OK){
    // UFS_Init is succeeded
}else{
    // UFS_Init is failed
    // Use UFS\_GetErrorString method to show error string
}
```

java JNI

```
int nRes =0;
UBioMiniJniSDK p = null;

//make instance
p = new BioMiniJniSDK();

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
}
```

UFS_Update

Enforces scanner module to update current connectivity of scanners.

```
UFS\_STATUS UFS_API UFS_Update();  
  
*java  
int UFS_Update();
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_NOT_INITIALIZED](#), [UFS_ERR_NO_LICENSE](#),
[UFS_ERR_LICENSE_NOT_MATCH](#), [UFS_ERR_LICENSE_EXPIRED](#),
[UFS_ERR_DEVICE_NUMBER_EXCEED](#)

See also

[UFS_Init](#), [UFS_Uninit](#)

Examples

Visual C++
UFS_STATUS ufs_res; ufs_res = UFS_Update(); if(ufs_res == UFS_OK) { // UFS_Update is succeeded } else { // UFS_Update is failed // Use UFS_GetErrorString function to show error string }

Visual Basic 6.0
Dim ufs_res As UFS_STATUS ufs_res = UFS_Update() If (ufs_res = UFS_STATUS.OK) Then ' UFS_Update is succeeded Else ' UFS_Update is failed ' Use UFS_GetErrorString function to show error string End If

java JNA

```
int ufs_res;

//load library(libScanner)

ufs_res = libScanner.UFS_Update();
if(ufs_res == libScanner.UFS_OK) {
    // UFS_Update is succeeded
} else {
    // UFS_Update is failed
    // Use UFS\_GetErrorString function to show error string
}
```

```
java JNI
int ufs_res;

//make class library instance(BioMiniJniSDK 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
}
```

UFS_Uninit

Un-initializes scanner module.

```
UFS\_STATUS UFS_API UFS_Uninit();  
  
*java  
int UFS_UnInit();
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_NOT_INITIALIZED](#), [UFS_ERR_NO_LICENSE](#),
[UFS_ERR_LICENSE_NOT_MATCH](#), [UFS_ERR_LICENSE_EXPIRED](#),
[UFS_ERR_DEVICE_NUMBER_EXCEED](#)

See also

[UFS_Init](#), [UFS_Update](#)

Examples

Visual C++
UFS_STATUS ufs_res; ufs_res = UFS_Uninit(); if(ufs_res == UFS_OK) { // UFS_Uninit is succeeded } else { // UFS_Uninit is failed // Use UFS_GetErrorString function to show error string }

Visual Basic 6.0
Dim ufs_res As UFS_STATUS ufs_res = UFS_Uninit() If (ufs_res = UFS_STATUS.OK) Then ' UFS_Update is succeeded Else ' UFS_Update is failed ' Use UFS_GetErrorString function to show error string End If

java JNA

```
int ufs_res;

//load library(libScanner)

ufs_res = libScanner.UFS_Uninit();
if(ufs_res == libScanner.UFS_OK) {
    // UFS_Uninit is succeeded
} else {
    // UFS_Uninit is failed
    // Use UFS\_GetErrorString function to show error string
}
```

```
java JNI
int ufs_res;

//make class library instance(BioMiniJniSDK p)

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

UFS_SetScannerCallback

Registers scanner callback function. Scanner callback is not working for every scanner model. Currently this function is working for Suprema SFR300-S(Ver.2) in windows 2000 / 2003 / XP only.

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

*java JNA
int UFS_SetScannerCallback(
    UFS_SCANNER_PROC pScannerProc,
    PointerByReference pParam
);

*java JNI
int UFS_SetScannerCallback(String nCallbackFunctionName);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

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

Remarks

This function is not supported on Visual Basic 6.0.

See also

[UFS_RemoveScannerCallback](#)

Examples

Visual C++
// Define scanner procedure

```

int UFS_CALLBACK ScannerProc(const char* szScannerID, int bSensorOn, void*
pParam)
{
    // ...
}

UFS_STATUS ufs_res;
void* pParam;

// Assign pParam, for example, application data

ufs_res = UFS_SetScannerCallback(ScannerProc, pParam);
if(ufs_res == UFS_OK) {
    // UFS_SetScannerCallback is succeeded
} else {
    // UFS_SetScannerCallback is failed
    // Use UFS\_GetErrorString function to show error string
}

```

java JNA
<pre> // Define scanner procedure UFScannerClass.UFS_SCANNER_PROC pScanProc = new UFScannerClass.UFS_SCANNER_PROC() { public int callback(String szScannerId,int bSensorOn,PointerByReference pParam) { //... return 1; } }; int ufs_res; PointerByReference refParam = new PointerByReference(); refParam.getPointer().setInt(0,1); // Assign pParam, for example, application data ufs_res= libScanner.UFS_SetScannerCallback(pScanProc,refParam); if(ufs_res==libScanner.UFS_OK){ // UFS_SetScannerCallback is succeeded }else{ // UFS_SetScannerCallback is failed // Use UFS_GetErrorString function to show error string } </pre>

java JNI

```
// Define scanner procedure

public void scannerCallback(char[] szScannerID, int bSensorOn) {
    // ...
}

// Set parameter, the name of call function for scanner event (USB Plug)

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
}
```

UFS_RemoveScannerCallback

Removes registered scanner callback function.

```
UFS_STATUS UFS_API UFS_RemoveScannerCallback();

*java
int UFS_RemoveScannerCallback();
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#)

Remarks

This function is not supported on Visual Basic 6.0.

See also

[UFS_SetScannerCallback](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;

ufs_res = UFS_RemoveScannerCallback();
if(ufs_res == UFS_OK) {
    // UFS_RemoveScannerCallback is succeeded
} else {
    // UFS_RemoveScannerCallback is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNA

```
int ufs_res;

//load library(libScanner)

ufs_res = libScanner.UFS_RemoveScannerCallback();
if(ufs_res == UFS_OK) {
    // UFS_RemoveScannerCallback is succeeded
} else {
    // UFS_RemoveScannerCallback is failed
```

```
// Use UFS\_GetErrorString function to show error string  
}
```

```
java JNI  
int ufs_res;  
  
//make class library instance(BioMiniJniSDK 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  
}
```

UFS_GetScannerNumber

Gets the number of scanners.

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

*java JNA
int UFS_GetScannerNumber(
    IntByReference pnScannerNumber
);

*java JNI
int UFS_GetScannerNumber(
    int[] pnScannerNumber
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#)

Parameters

pnScannerNumber	[out] Receives the number of scanners
-----------------	---------------------------------------

See also

[UFS_GetScannerHandle](#), [UFS_GetScannerIndex](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;
int nScannerNumber;

ufs_res = UFS_GetScannerNumber(&nScannerNumber);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerNumber is succeeded
} else {
    // UFS_GetScannerNumber is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim nScannerNumber As Long

ufs_res = UFS_GetScannerNumber(nScannerNumber)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetScannerNumber is succeeded
Else
    ' UFS_GetScannerNumber is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
IntByReference pScannerNumber;
int nNumber;

ufs_res = UFS_GetScannerNumber(pScannerNumber);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerNumber is succeeded
    nNumber = pScannerNumber.getValue();
} else {
    // UFS_GetScannerNumber is failed
    // Use UFS\_GetErrorString function to show error string
}f
```

java JNI

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

ufs_res = 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
}f
```

UFS_GetScannerHandle

Gets scanner handle using scanner index.

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

*java JNA
int UFS_GetScannerHandle(
    int nScannerIndex,
    PointerByReference phScanner
);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFS_GetScannerNumber](#), [UFS_GetScannerIndex](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;
int nScannerIndex;
HUFScanner hScanner;

// Set nScannerIndex to (0 ~ number of scanners - 1 )
// Number of scanner can be retrieved using UFS\_GetScannerNumber function
```

Suprema PC SDK 3.4.1

```
ufs_res = UFS_GetScannerHandle(nScannerIndex, &hScanner);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerHandle is succeeded
} else {
    // UFS_GetScannerHandle is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim nScannerIndex As Long
Dim hScanner As Long

' Set nScannerIndex to (0 ~ number of scanners - 1 )
' Number of scanner can be retrieved using UFS\_GetScannerNumber function

ufs_res = UFS_GetScannerHandle(nScannerIndex, hScanner)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetScannerHandle is succeeded
Else
    ' UFS_GetScannerHandle is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
int nScannerIndex;
Pointer hScanner;
PointerByReference refScanner = new PointerByReference();

// Set nScannerIndex to (0 ~ number of scanners - 1 )
// Number of scanner can be retrieved using UFS\_GetScannerNumber function
//load library(libScanner)

ufs_res = libScanner.UFS_GetScannerHandle(nScannerIndex, refScanner);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerHandle is succeeded
    hScanner = refScanner.getValue();
} else {
    // UFS_GetScannerHandle is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

```
int ufs_res;
long[] hScanner = new long[1];
int index = 0;
```

```
// Set nScannerIndex to (0 ~ number of scanners - 1 )
// Number of scanner can be retrieved using UFS\_GetScannerNumber function
//load library(libScanner)

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
}
```

UFS_GetScannerHandleByID

Gets scanner handle using scanner ID.

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

*java JNA
int UFS_GetScannerHandleByID(
    String szScannerID,
    PointerByReference phScanner
);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFS_GetScannerID](#)

Examples

Visual C++
<pre>UFS_STATUS ufs_res; char strID[64]; HUFScanner hScanner; // Assign scanner ID to strID // Scanner ID can be retrieved using UFS_GetScannerID function</pre>

```

ufs_res = UFS_GetScannerHandleByID(strID, &hScanner);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerHandleByID is succeeded
} else {
    // UFS_GetScannerHandleByID is failed
    // Use UFS\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

Dim ufs_res As UFS_STATUS
Dim strID As String
Dim hScanner As Long

' Assign scanner ID to strID
' Scanner ID can be retrieved using UFS\_GetScannerID function

ufs_res = UFS_GetScannerHandleByID(strID, hScanner)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetScannerHandleByID is succeeded
Else
    ' UFS_GetScannerHandleByID is failed
    ' Use UFS\_GetErrorString function to show error string
End If

```

java JNA

```

int ufs_res;
String strID;
Pointer hScanner;
PointerByReferece refScanner = new PointerByReference();

// Assign scanner ID to strID
// Scanner ID can be retrieved using UFS\_GetScannerID function

ufs_res = libScanner.UFS_GetScannerHandleByID(strID, refScanner);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerHandleByID is succeeded
    hScanner = refScanner.getValue();
} else {
    // UFS_GetScannerHandleByID is failed
    // Use UFS\_GetErrorString function to show error string
}

```

java JNI

```

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

// Assign scanner ID to strID

```

```
// Scanner ID can be retrieved using UFS\_GetScannerID function

ufs_res = p.UFS_GetScannerHandleByID(strID, hScanner );
if(ufs_res == UFS_OK) {
    // UFS_GetScannerHandleByID is succeeded
    // hScanner[0] is the handle
} else {
    // UFS_GetScannerHandleByID is failed
    // Use UFS\_GetErrorString function to show error string
}
```

UFS_GetScannerIndex

Gets scanner index assigned to scanner handle.

```
UFS_STATUS UFS_API UFS_GetScannerIndex(
    HUFScanner hScanner,
    int* pnScannerIndex
);

*java jNA
int UFS_GetScannerIndex(
    Pointer hScanner,
    IntByReference pnScannerIndex
);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFS_GetScannerNumber](#), [UFS_GetScannerHandle](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;
HUFScanner hScanner;
int nScannerIndex;

// Get hScanner handle

ufs_res = UFS_GetScannerIndex(hScanner, &nScannerIndex);
```

Suprema PC SDK 3.4.1

```
if(ufs_res == UFS_OK) {
    // UFS_GetScannerIndex is succeeded
} else {
    // UFS_GetScannerIndex is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nScannerIndex As Long

' Get hScanner handle

ufs_res = UFS_GetScannerIndex(hScanner, nScannerIndex)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetScannerIndex is succeeded
Else
    ' UFS_GetScannerIndex is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
int nScannerIndex;
IntByReference refIdx = new IntByReference();

// Get hScanner handle

ufs_res = libScanner.UFS_GetScannerIndex(hScanner, refIdx);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerIndex is succeeded
    nScannerIndex = refIdx.getValue();
} else {
    // UFS_GetScannerIndex is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNA

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

// Get hScanner handle

ufs_res = p.UFS_GetScannerIndex(hScanner[0], nScannerIndex );
if(ufs_res == UFS_OK) {
```

```
// UFS_GetScannerIndex is succeeded  
// nScanndeIndex[0] is the index  
} else {  
    // UFS_GetScannerIndex is failed  
    // Use UFS\_GetErrorString function to show error string  
}
```

UFS_GetScannerID

Gets scanner ID assigned to scanner handle.

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

*java JNA
int UFS_GetScannerID(
    Pointer hScanner,
    byte[] pScannerID
);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

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.

See also

[UFS_GetScannerHandleByID](#)

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; char strID[64]; // Should be larger than 33 bytes</pre>

```
// Get hScanner handle

ufs_res = UFS_GetScannerID(hScanner, strID);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerID is succeeded
} else {
    // UFS_GetScannerID is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim strID As String // In Visual basic, string is assigned from the SDK

' Get hScanner handle

ufs_res = UFS_GetScannerID(hScanner, strID)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetScannerID is succeeded
Else
    ' UFS_GetScannerID is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
String strID; // Should be larger than 33 bytes

//load library(libScanner)

// Get hScanner handle

ufs_res = libScanner.UFS_GetScannerID(hScanner, strID);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_GetScannerID is succeeded
} else {
    // UFS_GetScannerID is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNA

```
int ufs_res;
long[] hScanner = new long[1];
byte[] strID = new byte[128]; // Should be larger than 33 bytes

// make class library instance(BioMiniJniSDK p)
```

```
// Get hScanner handle

ufs_res = p.UFS_GetScannerID(hScanner[0], 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
}
```

UFS_GetScannerType

Gets scanner type assigned to scanner handle.

```
UFS_STATUS UFS_API UFS_GetScannerType(
    HUFScanner hScanner,
    int* pnScannerType
);

*java JNA
int UFS_GetScannerType(
    Pointer hScanner,
    IntByReference pnScannerType
);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

Examples

Visual C++

```
UFS_STATUS ufs_res;
HUFScanner hScanner;
int nScannerType;

// Get hScanner handle

ufs_res = UFS_GetScannerType(hScanner, &nScannerType);
if(ufs_res == UFS_OK) {
    // UFS_GetScannerType is succeeded
} else {
    // UFS_GetScannerType is failed
```

```
// Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nScannerType As Long

' Get hScanner handle

ufs_res = UFS_GetScannerType(hScanner, nScannerType)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetScannerType is succeeded
Else
    ' UFS_GetScannerType is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
IntByReference refType = new IntByReference();
int nScannerType;

//load library(libScanner)

// Get hScanner handle

ufs_res = libScanner.UFS_GetScannerType(hScanner, refType);
if (ufs_res == UFS_OK) {
    // UFS_GetScannerType is succeeded
    nScannerType = refType.getValue();
} else {
    // UFS_GetScannerType is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

ufs_res = p.UFS_GetScannerType(hScanner[0], 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  
}
```

UFS_GetParameter

Gets parameter value.

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

*java JNA
int UFS_GetParameter(
    Pointer hScanner,
    int nParam,
    PointerByReference pValue);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

hScanner	[in] Handle to the scanner object
nParam	[in] Parameter type; one of parameters
pValue	[out] Receives parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

See also

[UFS_SetParameter](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;
HUFScanner hScanner;
```

```

int nValue;
char strSerial[64];

// Get hScanner handle

// Get timeout
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_TIMEOUT, &nValue);
// Error handling routine is omitted

// Get brightness
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_BRIGHTNESS, &nValue);
// Error handling routine is omitted

// Get sensitivity
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_SENSITIVITY, &nValue);
// Error handling routine is omitted

// Get serial
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_SERIAL, strSerial);
// Error handling routine is omitted

// Get detect core
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_DETECT_CORE, &nValue);
// Error handling routine is omitted

// Get template size
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_TEMPLATE_SIZE, &nValue);
// Error handling routine is omitted

// Get use SIF
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted

```

Visual Basic 6.0

```

Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nValue As Long
Dim strSerial As String

' Get hScanner handle

' Get timeout
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_TIMEOUT, nValue)
' Error handling routine is omitted

' Get brightness
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM_BRIGHTNESS, nValue)
' Error handling routine is omitted

```

```
' Get sensitivity
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM.SENSITIVITY, nValue)
' Error handling routine is omitted

' Get serial
' Caution: for get serial, UFS_GetParameter_B is used
ufs_res = UFS_GetParameter_B(hScanner, UFS_PARAM.SERIAL, strSerial)
' Error handling routine is omitted

' Get detect core
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM.DETECT_CORE, nValue)
' Error handling routine is omitted

' Get template size
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM.TEMPLATE_SIZE, nValue)
' Error handling routine is omitted

' Get use SIF
ufs_res = UFS_GetParameter(hScanner, UFS_PARAM.USE_SIF, nValue)
' Error handling routine is omitted
```

```
java JNA
int ufs_res;
Pointer hScanner;
int nValue;
IntByReference refValue = new IntByReference();

//load library(libScanner)

// Get hScanner handle

// Get timeout
ufs_res = libScanner.UFS_GetParameter(hScanner, UFS_PARAM_TIMEOUT,
refValue );
nValue=refValue.getValue();
// Error handling routine is omitted

// Get brightness
ufs_res = libScanner.UFS_GetParameter(hScanner, UFS_PARAM_BRIGHTNESS,
refValue );
// Error handling routine is omitted

// Get sensitivity
ufs_res = libScanner.UFS_GetParameter(hScanner, UFS_PARAM_SENSITIVITY,
refValue );
// Error handling routine is omitted

// Get detect core
ufs_res = libScanner.UFS_GetParameter(hScanner, UFS_PARAM_DETECT_CORE,
```

```

refValue );
// Error handling routine is omitted

// Get template size
ufs_res = libScanner.UFS_GetParameter(hScanner, UFS_PARAM_TEMPLATE_SIZE,
refValue );
// Error handling routine is omitted

// Get use SIF
ufs_res = libScanner.UFS_GetParameter(hScanner, UFS_PARAM_USE_SIF, refValue );
// Error handling routine is omitted

```

```

java JNI
int ufs_res;
long[] hScanner = new long[1];
int[] nValue = new int[1];

//make class library instance(BioMiniJniSDK p)

// Get hScanner handle

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

// Get brightness
ufs_res = p.UFS_GetParameter(hScanner[0], p.UFS_PARAM_BRIGHTNESS, nValue );
// Error handling routine is omitted

// Get sensitivity
ufs_res = p.UFS_GetParameter(hScanner[0], p.UFS_PARAM_SENSITIVITY, nValue );
// Error handling routine is omitted

// Get detect core
ufs_res = p.UFS_GetParameter(hScanner[0], p.UFS_PARAM_DETECT_CORE,
nValue );
// Error handling routine is omitted

// Get template size
ufs_res = p.UFS_GetParameter(hScanner[0], p.UFS_PARAM_TEMPLATE_SIZE,
nValue );
// Error handling routine is omitted

// Get use SIF
ufs_res = p.UFS_GetParameter(hScanner[0], p.UFS_PARAM_USE_SIF, nValue );
// Error handling routine is omitted

```

UFS_SetParameter

Sets parameter value.

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

*java JNA
int UFS_SetParameter(
    Pointer hScanner,
    int nParam,
    IntByReference pValue
);

*java JNI
int UFS_SetParameter(
    long hScanner,
    int nParam,
    int[] pValue
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

hScanner	[in] Handle to the scanner object
nParam	[in] Parameter type; one of parameters
pValue	[in] Pointer to parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

See also

[UFS_GetParameter](#)

Examples

Visual C++
UFS_STATUS ufs_res;

```

HUFScanner hScanner;
int nValue;

// Get hScanner handle

// Set timeout to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT, &nValue);
// Error handling routine is omitted

// Set brightness to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_BRIGHTNESS, &nValue);
// Error handling routine is omitted

// Set sensitivity to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_SENSITIVITY, &nValue);
// Error handling routine is omitted

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

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

// Set use SIF to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted

```

Visual Basic 6.0

```

Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nValue As Long

' Get hScanner handle

' Set timeout to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT, nValue)
' Error handling routine is omitted

' Set brightness to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_BRIGHTNESS, nValue)
' Error handling routine is omitted

' Set sensitivity to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_SENSITIVITY, nValue)
' Error handling routine is omitted

' Set detect core to nValue

```

```

ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_DETECT_CORE, nValue)
' Error handling routine is omitted

' Set template size to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_TEMPLATE_SIZE, nValue)
' Error handling routine is omitted

' Set use SIF to nValue
ufs_res = UFS_SetParameter(hScanner, UFS_PARAM_USE_SIF, nValue)
' Error handling routine is omitted

```

```

java JNI
int ufs_res;
Pointer hScanner;
IntByReference refValue = new IntByReference();

//load library(libScanner)

// Get hScanner handle

// Set timeout to nValue
refValue.setValue(5);
ufs_res = libScanner.UFS_SetParameter(hScanner, UFS_PARAM_TIMEOUT,
revValue);
// Error handling routine is omitted

// Set brightness to nValue
refValue.setValue(100);
ufs_res = libScanner.UFS_SetParameter(hScanner, UFS_PARAM_BRIGHTNESS,
revValue);
// Error handling routine is omitted

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

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

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

// Set use SIF to nValue
ufs_res = libScanner.UFS_SetParameter(hScanner, UFS_PARAM_USE_SIF, revValue);

```

```
// Error handling routine is omitted
```

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

//make class library instance(BioMiniJniSDK p)

// Get hScanner handle

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

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

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

// Set detect core to nValue
ufs_res = p.UFS_SetParameter(hScanner[0], p.UFS_PARAM_DETECT_CORE, nValue);
// Error handling routine is omitted

// Set template size to nValue
ufs_res = p.UFS_SetParameter(hScanner[0], p.UFS_PARAM_TEMPLATE_SIZE,
nValue);
// Error handling routine is omitted

// Set use SIF to nValue
ufs_res = p.UFS_SetParameter(hScanner[0], p.UFS_PARAM_USE_SIF, nValue);
// Error handling routine is omitted
```

UFS_IsSensorOn

Checks the scanner is connected or not.

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

*java JNA
int UFS_IsSensorOn(
    Pointer hScanner,
    IntByReference pbSensorOn);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

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

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; int bSensorOn; // Get hScanner handle ufs_res = UFS_IsSensorOn(hScanner, &bSensorOn); if(ufs_res == UFS_OK) { // UFS_IsSensorOn is succeeded } else {</pre>

```
// UFS_IsSensorOn is failed
// Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim bSensorOn As Long

' Get hScanner handle

ufs_res = UFS_IsSensorOn(hScanner, bSensorOn)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_IsSensorOn is succeeded
Else
    ' UFS_IsSensorOn is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
IntByReference bSensorOn = new IntByReference();

//load library

// Get hScanner handle

ufs_res = libScanner.UFS_IsSensorOn(hScanner, bSensorOn);
if (ufs_res == libScanner.UFS_OK) {
    // UFS_IsSensorOn is succeeded
} else {
    // UFS_IsSensorOn is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

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

//make class library instance(BioMiniJniSDK p)

// Get hScanner handle

ufs_res = p.UFS_IsSensorOn(hScanner[0], bSensorOn);
if (ufs_res == p.UFS_OK) {
    // UFS_IsSensorOn is succeeded
```

```
    } else {
        // UFS_IsSensorOn is failed
        // Use UFS\_GetErrorString function to show error string
    }
```

UFS_IsFingerOn

Checks a finger is placed on the scanner or not.

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

*java JNA

int UFS_IsFingerOn(
    Pointer hScanner,
    IntByReference pbFingerOn)

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

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

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; int bFingerOn; // Get hScanner handle ufs_res = UFS_IsFingerOn(hScanner, &bFingerOn); if(ufs_res == UFS_OK) { // UFS_IsFingerOn is succeeded } else {</pre>

Suprema PC SDK 3.4.1

```
// UFS_IsFingerOn is failed  
// Use UFS\_GetErrorString function to show error string  
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS  
Dim hScanner As Long  
Dim bFingerOn As Long  
  
' Get hScanner handle  
  
ufs_res = UFS_IsFingerOn(hScanner, bFingerOn)  
If (ufs_res = UFS_STATUS.OK) Then  
    ' UFS_IsFingerOn is succeeded  
Else  
    ' UFS_IsFingerOn is failed  
    ' Use UFS\_GetErrorString function to show error string  
End If
```

java JNA

```
int ufs_res;  
Pointer hScanner;  
IntByReference bFingerOn = new IntByReference();  
  
//load library(libScanner)  
  
// Get hScanner handle  
  
ufs_res = libScanner.UFS_IsFingerOn(hScanner, bFingerOn);  
if (ufs_res == libScanner.UFS_OK) {  
    // UFS_IsFingerOn is succeeded  
} else {  
    // UFS_IsFingerOn is failed  
    // Use UFS\_GetErrorString function to show error string  
}
```

java JNI

```
int ufs_res;  
long[] hScanner = new long[1];  
int[] bFingerOn = new int[1];  
  
// make class library instance(BioMiniJniSDK p)  
  
// Get hScanner handle  
  
ufs_res = p.UFS_IsFingerOn(hScanner[0], bFingerOn);  
if (ufs_res == p.UFS_OK) {  
    // UFS_IsSensorOn is succeeded
```

```
    } else {
        // UFS_IsSensorOn is failed
        // Use UFS\_GetErrorString function to show error string
    }
```

UFS_CaptureSingleImage

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

```
UFS\_STATUS UFS_API UFS_CaptureSingleImage(  
    HUFScanner hScanner,  
);  
  
*java JNA  
int UFS_CaptureSingleImage(Pointer hScanner);  
  
*java JNI  
int UFS_CaptureSingleImage(long hScanner);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#),
[UFS_ERR_CAPTURE_RUNNING](#)

Parameters

hScanner	[in] Handle to the scanner object
----------	-----------------------------------

See also

[UFS_IsCapturing](#), [UFS_GetCaptureImageBufferInfo](#), [UFS_GetCaptureImageBuffer](#),
[UFS_DrawCaptureImageBuffer](#), [UFS_SaveCaptureImageBufferToBMP](#),
[UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;  
HUFScanner hScanner;  
  
// Get hScanner handle  
  
ufs_res = UFS_CaptureSingleImage(hScanner);  
if (ufs_res == UFS_OK) {  
    // UFS_CaptureSingleImage is succeeded  
} else {  
    // UFS_CaptureSingleImage is failed  
    // Use UFS\_GetErrorString function to show error string  
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long

' Get hScanner handle

ufs_res = UFS_CaptureSingleImage(hScanner)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_CaptureSingleImage is succeeded
Else
    ' UFS_CaptureSingleImage is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;

//load library(libScanner)

// Get hScanner handle

ufs_res = libScanner.UFS_CaptureSingleImage(hScanner);
if (ufs_res == libScanner.UFS_OK) {
    // UFS_CaptureSingleImage is succeeded
} else {
    // UFS_CaptureSingleImage is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

ufs_res = p.UFS_CaptureSingleImage(hScanner[0]);
if (ufs_res == p.UFS_OK) {
    // UFS_CaptureSingleImage is succeeded
} else {
    // UFS_CaptureSingleImage is failed
    // Use UFS\_GetErrorString function to show error string
}
```

UFS_StartCapturing

Starts capturing. Currently this function is working for Suprema SFR300-S(Ver.2) only.

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

*java JNA
int UFS_StartCapturing(
    Pointer hScanner,
    UFS_CAPTURE_PROC pCaptureProc,
    PointerByReference pParam);

*java JNI
int UFS_StartCapturing(
    long hScanner,
    String nCallbackFunctionName
);
```

Possible return values

UFS_OK, UFS_ERROR, UFS_ERR_NOT_SUPPORTED,
UFS_ERR_INVALID_PARAMETERS, UFS_ERR_CAPTURE_RUNNING

Parameters

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

Remarks

This function is not supported on Visual Basic 6.0.

Supported scanners

Suprema SFR300-S(Ver.2)

See also

[UFS_IsCapturing](#), [UFS_AbortCapturing](#), [UFS_GetCaptureImageBufferInfo](#),
[UFS_GetCaptureImageBuffer](#), [UFS_DrawCaptureImageBuffer](#),
[UFS_SaveCaptureImageBufferToBMP](#), [UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++

```
// 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;
void* pParam;

// Get hScanner handle

// Assign pParam, for example, application data

ufs_res = UFS_StartCapturing(hScanner, CaptureProc, pParam);
if(ufs_res == UFS_OK) {
    // UFS_StartCapturing is succeeded
} else {
    // UFS_StartCapturing is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNA

```
// Define capture procedure

UFScannerClass.UFS_CAPTURE_PROC pCaptureProc = new
UFScannerClass.UFS_CAPTURE_PROC()
{
    public int callback(Pointer hScanner, int bFingerOn, Pointer pImage, int nWidth, int
nHeight, int nResolution, PointerByReference pParam)
    {
        // ...
        //drawCurrentFingerImage();
        return 1;
    }
};

int ufs_res;
Pointer hScanner;
PointerByReference pParam = new PointerByReference();
```

```
//load library(libScanner)

// Get hScanner handle

// Assign pParam, for example, application data

ufs_res = libScanner.UFS_StartCapturing(hScanner, CaptureProc, pParam);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_StartCapturing is succeeded
} else {
    // UFS_StartCapturing is failed
    // Use UFS\_GetErrorString function to show error string
}
```

```
java JNI
// 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(BioMiniJniSDK p)

// Get hScanner handle

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

ufs_res = p.UFS_StartCapturing(hScanner[0], "captureCallback");
if(ufs_res == p.UFS_OK) {
    // UFS_StartCapturing is succeeded
} else {
    // UFS_StartCapturing is failed
    // Use UFS\_GetErrorString function to show error string
}
```

UFS_IsCapturing

Checks if the specified scanner is running capturing which is started by [UFS_CaptureSingleImage](#) or [UFS_StartCapturing](#).

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

*java JNA
int UFS_IsCapturing(
    Pointer hScanner,
    IntByReference pbCapturing
);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

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

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#)

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; int bCapturing; // Get hScanner handle</pre>

Suprema PC SDK 3.4.1

```
ufs_res = UFS_IsCapturing(hScanner, &bCapturing);
if(ufs_res == UFS_OK) {
    // UFS_IsCapturing is succeeded
} else {
    // UFS_IsCapturing is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim bCapturing As Long

' Get hScanner handle

ufs_res = UFS_IsSensorOn(hScanner, bCapturing)
If(ufs_res = UFS_STATUS.OK) Then
    ' UFS_IsCapturing is succeeded
Else
    ' UFS_IsCapturing is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
IntByReference bCapturing = new IntByReference();

//load libaray(libScanner)

// Get hScanner handle

ufs_res = libScanner.UFS_IsCapturing(hScanner, bCapturing);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_IsCapturing is succeeded
} else {
    // UFS_IsCapturing is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

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

// make class library instance(BioMiniJniSDK p)
```

```
// Get hScanner handle

ufs_res = p.UFS_IsCapturing(hScanner[0], bCapturing);
if(ufs_res == p.UFS_OK) {
    // UFS_IsSensorOn is succeeded
} else {
    // UFS_IsSensorOn is failed
    // Use UFS\_GetErrorString function to show error string
}
```

UFS_AbortCapturing

Aborts capturing which is started by [UFS_CaptureSingleImage](#) or [UFS_StartCapturing](#).

```
UFS\_STATUS UFS_API UFS_AbortCapturing(  
    HUFScanner hScanner,  
);  
  
*java JNA  
int UFS_AbortCapturing(Pointer hScanner);  
  
*java JNI  
int UFS_AbortCapturing(long hScanner);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_NOT_SUPPORTED](#),
[UFS_ERR_INVALID_PARAMETERS](#)

Parameters

hScanner	[in] Handle to the scanner object
----------	-----------------------------------

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_IsCapturing](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;  
HUFScanner hScanner;  
  
// Get hScanner handle  
  
// Start capturing  
  
ufs_res = UFS_AbortCapturing(hScanner);  
if (ufs_res == UFS_OK) {  
    // UFS_AbortCapturing is succeeded  
} else {  
    // UFS_AbortCapturing is failed  
    // Use UFS\_GetErrorString function to show error string  
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long

' Get hScanner handle

' Start capturing

ufs_res = UFS_AbortCapturing(hScanner)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_AbortCapturing is succeeded
Else
    ' UFS_AbortCapturing is failed
    ' Use UFS_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;

//load library(libScanner)

// Get hScanner handle

// Start capturing

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

java JNI

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

ufs_res = p.UFS_AbortCapturing(hScanner[0]);
if (ufs_res == p.UFS_OK) {
    // UFS_AbortCapturing is succeeded
} else {
    // UFS_AbortCapturing is failed
```

```
    } // Use UFS\_GetErrorString function to show error string
```

UFS_Extract

Extracts a template from the stored image buffer which is acquired using [UFS_CaptureSingleImage\(\)](#) or [UFS_StartCapturing\(\)](#).

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

*java JNA
int UFS_Extract(
    Pointer hScanner,
    byte[] pTemplate,
    IntByReference pnTemplateSize,
    IntByReference pnEnrollQuality);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_LICENSE_EXPIRED](#),
[UFS_ERR_LICENSE_NOT_MATCH](#), [UFS_ERR_INVALID_PARAMETERS](#),
[UFS_ERR_NOT_GOOD_IMAGE](#), [UFS_ERR_EXTRACTION_FAILED](#),
[UFS_ERR_CORE_NOT_DETECTED](#), [UFS_ERR_CORE_TO_LEFT](#),
[UFS_ERR_CORE_TO_LEFT_TOP](#), [UFS_ERR_CORE_TO_TOP](#),
[UFS_ERR_CORE_TO_RIGHT_TOP](#), [UFS_ERR_CORE_TO_RIGHT](#),
[UFS_ERR_CORE_TO_RIGHT_BOTTOM](#), [UFS_ERR_CORE_TO_BOTTOM](#),
[UFS_ERR_CORE_TO_LEFT_BOTTOM](#)

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.
-----------------	---

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#)

Examples

Visual C++

```
// Template size can be controlled by using UFS\_SetParameter function
// Default value is 384 bytes
#define MAX_TEMPLATE_SIZE 384

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

// Get hScanner handle

ufs_res = UFS_Extract(hScanner, Template, &TemplateSize, &nEnrollQuality);
if(ufs_res == UFS_OK) {
    // UFS_Extract is succeeded
} else {
    // UFS_Extract is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
' Template size can be controlled by using UFS\_SetParameter function
' Default value is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateSize As Long
Dim EnrollQuality As Long

' Get hScanner handle

ufs_res = UFS_Extract(hScanner, Template(0), TemplateSize, EnrollQuality)
If(ufs_res = UFS_STATUS.OK) Then
    'UFS_Extract is succeeded
```

```

Else
    ' UFS_Extract is failed
    ' Use UFS\_GetErrorString function to show error string
End If

```

```

java JNA
// Template size can be controlled by using UFS\_SetParameter function
// Default value is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufs_res;
Pointer hScanner;
byte[] Template = new byte[MAX_TEMPLATE_SIZE];
IntByReference TemplateSize = new IntByReference();
IntByReference nEnrollQuality =new IntByReference();

//load library(libScanner)

// Get hScanner handle

ufs_res = libScanner.UFS_Extract(hScanner, Template, TemplateSize, nEnrollQuality);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_Extract is succeeded
} else {
    // UFS_Extract is failed
    // Use UFS\_GetErrorString function to show error string
}

```

```

java JNI
// Template size can be controlled by using UFS\_SetParameter function
// Default value is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

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(BioMiniJniSDK p)

// Get hScanner handle

ufs_res = p.UFS_Extract(hScanner[0], Template, TemplateSize, TemplateQuality);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_Extract is succeeded
} else {
    // UFS_Extract is failed
}

```

```
// Use UFS\_GetErrorString function to show error string  
}
```

UFS_SetEncryptionKey

Sets encryption key.

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

*java JNA
int UFS_SetEncryptionKey(
    Pointer hScanner,
    String pKey
);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFS_EncryptTemplate](#), [UFS_DecryptTemplate](#)

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; unsigned char UserKey[32]; // Get hScanner handle</pre>

Suprema PC SDK 3.4.1

```
// Generate 32 byte encryption key to UserKey

ufs_res = UFS_SetEncryptionKey(hScanner, UserKey);
if(ufs_res == UFS_OK) {
    // UFS_SetEncryptionKey is succeeded
} else {
    // UFS_SetEncryptionKey is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim UserKey(32 - 1) As Byte

' Get hScanner handle

' Generate 32 byte encryption key to UserKey

ufs_res = UFS_SetEncryptionKey(hScanner, UserKey)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_SetEncryptionKey is succeeded
Else
    ' UFS_SetEncryptionKey is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
String UserKey;

//load library(libScanner)

// Get hScanner handle

// Generate 32 byte encryption key to UserKey

ufs_res = libScanner.UFS_SetEncryptionKey(hScanner, UserKey);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_SetEncryptionKey is succeeded
} else {
    // UFS_SetEncryptionKey is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

```
int ufs_res;
```

```
long[] hScanner = new long[1];
byte[] UserKey = new byte[32];

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

// Generate 32 byte encryption key to UserKey

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

UFS_EncryptTemplate

Encrypts template.

```
UFS\_STATUS UFS_API UFS_EncryptTemplate(
    HUFScanner hScanner,
    unsigned char* pTemplateInput,
    int nTemplateInputSize,
    unsigned char* pTemplateOutput,
    int* pnTemplateOutputSize
);

*java JNA
int UFS_EncryptTemplate(
    Pointer hScanner,
    byte[] pTemplateInput,
    int nTemplateInputSize,
    byte[] pTemplateOutput,
    IntByReference pnTemplateOutputSize
);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFS_SetEncryptionKey](#), [UFS_DecryptTemplate](#)

Examples

Visual C++

```
// Assume template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384

UFS_STATUS ufs_res;
HUFScanner hScanner;
unsigned char TemplateInput[MAX_TEMPLATE_SIZE];
unsigned char TemplateOutput[MAX_TEMPLATE_SIZE];
int TemplateInputSize;
int TemplateOutputSize;

// Get 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 is succeeded
} else {
    // UFS_EncryptTemplate is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim TemplateInput(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateOutput(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateInputSize As Long
Dim TemplateOutputSize As Long

' Get 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(0), TemplateInputSize,
TemplateOutput(0), TemplateOutputSize)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_EncryptTemplate is succeeded
Else
    ' UFS_EncryptTemplate is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

```
java JNA
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE 384

int ufs_res;
Pointer hScanner;
byte[] TemplateInput = new byte[MAX_TEMPLATE_SIZE];
byte[] TemplateOutput = new byte[MAX_TEMPLATE_SIZE];
int TemplateInputSize;
IntByReference TemplateOutputSize = new IntByReference();;
//load library\(libScanner\)

// Get hScanner handle

// Get an input template to encrypt, TemplateInput and TemplateInputSize

// Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE;

ufs_res = libScanner.UFS_EncryptTemplate(hScanner, TemplateInput,
TemplateInputSize, TemplateOutput, TemplateOutputSize);
if (ufs_res == libScanner.UFS_OK) {
    // UFS_EncryptTemplate is succeeded
} else {
    // UFS_EncryptTemplate is failed
    // Use UFS\_GetErrorString function to show error string
}
```

```
java JNI
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE 384

int ufs_res;
Pointer hScanner;
byte[] TemplateInput = new byte[MAX_TEMPLATE_SIZE];
byte[] TemplateOutput = new byte[MAX_TEMPLATE_SIZE];
```

```
int TemplateInputSize;  
  
// make class library instance(BioMiniJniSDK p)  
  
// Get hScanner handle  
  
// Get an input template to encrypt, TemplateInput and TemplateInputSize  
  
// Set output template buffer size  
int[] TemplateoutputSize = new int[MAX_TEMPLATE_SIZE];  
  
ufs_res = p.UFS_EncryptTemplate(hScanner[0], TemplateInput, TemplateInputSize,  
TemplateOutput, TemplateOutputSize);  
if(ufs_res == p.UFS_OK) {  
    // UFS_EncryptTemplate is succeeded  
} else {  
    // UFS_EncryptTemplate is failed  
    // Use UFS_GetErrorString function to show error string  
}
```

UFS_DecryptTemplate

Decrypts template.

```
UFS\_STATUS UFS_API UFS_DecryptTemplate(
    HUFScanner hScanner,
    unsigned char* pTemplateInput,
    int nTemplateInputSize,
    unsigned char* pTemplateOutput,
    int* pnTemplateOutputSize
);

*java JNA
int UFS_DecryptTemplate(
    Pointer hScanner,
    byte[] pTemplateInput,
    int nTemplateInputSize,
    byte[] pTemplateOutput,
    IntByReference pnTemplateOutputSize
);

*java JNI
int UFS_DecryptTemplate(
    long hScanner,
    byte[] pTemplateInput,
    int nTemplateInputSize,
    byte[] pTemplateOutput,
    int[] pnTemplateOutputSize
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFS_SetEncryptionKey](#), [UFS_EncryptTemplate](#)

Examples

Visual C++

```
// Assume template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384

UFS_STATUS ufs_res;
HUFScanner hScanner;
unsigned char TemplateInput[MAX_TEMPLATE_SIZE];
unsigned char TemplateOutput[MAX_TEMPLATE_SIZE];
int TemplateInputSize;
int TemplateOutputSize;

// Get 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 is succeeded
} else {
    // UFS_DecryptTemplate is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim TemplateInput(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateOutput(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateInputSize As Long
Dim TemplateOutputSize As Long

' Get hScanner handle

' Get an encrypted template, TemplateInput and TemplateInputSize

' Set output template buffer size
```

```
TemplateoutputSize = MAX_TEMPLATE_SIZE

ufs_res = UFS_DecryptTemplate(hScanner, TemplateInput(0), TemplateInputSize,
TemplateOutput(0), TemplateOutputSize)
If (ufs_res == UFS_STATUS.OK) Then
    ' UFS_DecryptTemplate is succeeded
Else
    ' UFS_DecryptTemplate is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

```
java JNA
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE 384;

int ufs_res;
Pointer hScanner;
byte[] TemplateInput = new byte[MAX_TEMPLATE_SIZE];
byte[] TemplateOutput = new byte[MAX_TEMPLATE_SIZE];
int TemplateInputSize;
IntByReference TemplateOutputSize = new IntByReference();

//load libray(libScanner)

// Get hScanner handle

// Get an encrypted template, TemplateInput and TemplateInputSize

// Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE;

ufs_res = libScanner.UFS_DecryptTemplate(hScanner, TemplateInput,
TemplateInputSize, TemplateOutput, TemplateOutputSize);
if (ufs_res == libScanner.UFS_OK) {
    // UFS_DecryptTemplate is succeeded
} else {
    // UFS_DecryptTemplate is failed
    // Use UFS\_GetErrorString function to show error string
}
```

```
java JNI
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE 384;

int ufs_res;
Pointer hScanner;
byte[] TemplateInput = new byte[MAX_TEMPLATE_SIZE];
byte[] TemplateOutput = new byte[MAX_TEMPLATE_SIZE];
```

```
int TemplateInputSize;  
  
//load library(libScanner)  
  
// Get hScanner handle  
  
// Get an encrypted template, TemplateInput and TemplateInputSize  
  
// Set output template buffer size  
int[] TemplateoutputSize = new int[MAX_TEMPLATE_SIZE];  
  
ufs_res = p.UFS_DecryptTemplate(hScanner[0], TemplateInput, TemplateInputSize,  
TemplateOutput, TemplateOutputSize);  
if(ufs_res == p.UFS_OK) {  
    // UFS_DecryptTemplate is succeeded  
} else {  
    // UFS_DecryptTemplate is failed  
    // Use UFS_GetErrorString function to show error string  
}
```

UFS_GetCaptureImageBufferInfo

Gets the information of the capture image buffer.

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

*java JNI
int UFS_GetCaptureImageBufferInfo(
    Pointer hScanner,
    IntByReference pnWidth,
    IntByReference pnHeight,
    IntByReference pnResolution);

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_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

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_GetCaptureImageBuffer](#),
[UFS_DrawCaptureImageBuffer](#), [UFS_SaveCaptureImageBufferToBMP](#),
[UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++

```

UFS_STATUS ufs_res;
HUFScanner hScanner;
int nWidth;
int nHeight;
int nResolution;

// Get hScanner handle

ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, &nWidth, &nHeight,
&nResolution);
if(ufs_res == UFS_OK) {
    // UFS_GetCaptureImageBufferInfo is succeeded
} else {
    // UFS_GetCaptureImageBufferInfo is failed
    // Use UFS\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nWidth As Long
Dim nHeight As Long
Dim nResolution As Long

' Get hScanner handle

ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, nWidth, nHeight, nResolution)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetCaptureImageBufferInfo is succeeded
Else
    ' UFS_GetCaptureImageBufferInfo is failed
    ' Use UFS\_GetErrorString function to show error string
End If

```

java JNA

```

int ufs_res;
Pointer hScanner;
IntByReference nWidth = new IntByReference();
IntByReference nHeight = new IntByReference();
IntByReference nResolution = new IntByReference();

//load library(libScanner)

// Get hScanner handle

ufs_res = libScanner.UFS_GetCaptureImageBufferInfo(hScanner, nWidth, nHeight,

```

```
nResolution);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_GetCaptureImageBufferInfo is succeeded
} else {
    // UFS_GetCaptureImageBufferInfo is failed
    // Use UFS\_GetErrorString function to show error string
}
```

```
java JNI
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(BioMiniJniSDK p)

// Get hScanner handle

ufs_res = p.UFS_GetCaptureImageBufferInfo(hScanner[0], 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
}
```

UFS_GetCaptureImageBuffer

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

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

*java JNA
int UFS_GetCaptureImageBuffer(
    Pointer hScanner,
    byte[] pImageData
);

*java JNI
int UFS_GetCaptureImageBuffer(
    long hScanner,
    byte[] pImageData
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

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

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_GetCaptureImageBufferInfo](#),
[UFS_DrawCaptureImageBuffer](#), [UFS_SaveCaptureImageBufferToBMP](#),
[UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++
UFS_STATUS ufs_res; HUFScanner hScanner; int nWidth; int nHeight;

```
int nResolution;
unsigned char* pImageData

// Get hScanner handle

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

// 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 is succeeded
} else {
    // UFS_GetCaptureImageBuffer is failed
    // Use UFS_GetErrorString function to show error string
}

// Free image buffer after usage
free(pImageData)
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nWidth As Long
Dim nHeight As Long
Dim nResolution As Long
Dim ImageData As Byte

' Get hScanner handle

' Get capture image buffer information
ufs_res = UFS_GetCaptureImageBufferInfo(hScanner, nWidth, nHeight, nResolution)
' Error handling routine is omitted

' Allocate image buffer
ReDim ImageData(nWidth * nHeight - 1) As Byte

ufs_res = UFS_GetCaptureImageBuffer(hScanner, ImageData(0))
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetCaptureImageBuffer is succeeded
Else
    ' UFS_GetCaptureImageBuffer is failed
    ' Use UFS_GetErrorString function to show error string
End If
```

```

java JNA
int ufs_res;
Pointer hScanner;
IntByReference nWidth = new IntByReference();
IntByReference nHeight= new IntByReference();
IntByReference nResolution= new IntByReference();

// Get hScanner handle

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

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

ufs_res = libScanner.UFS_GetCaptureImageBuffer(hScanner, pImageData);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_GetCaptureImageBuffer is succeeded
} else {
    // UFS_GetCaptureImageBuffer is failed
    // Use UFS\_GetErrorString function to show error string
}

```

```

java JNI
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[0], nWidth, nHeight,
nResolution);
// Error handling routine is omitted

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

ufs_res = p.UFS_GetCaptureImageBuffer(hScanner[0], pImageData);
if(ufs_res == p.UFS_OK) {
    // UFS_GetCaptureImageBuffer is succeeded
} else {
    // UFS_GetCaptureImageBuffer is failed
    // Use UFS\_GetErrorString function to show error string
}

```

}

UFS_DrawCaptureImageBuffer

Draws the fingerprint image which is acquired using [UFS_CaptureSingleImage\(\)](#) or [UFS_StartCapturing\(\)](#).

This function is not supported on java.

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

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

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 or not

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_GetCaptureImageBufferInfo](#),
[UFS_GetCaptureImageBuffer](#), [UFS_SaveCaptureImageBufferToBMP](#),
[UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++

UFS_STATUS ufs_res; HUFScanner hScanner;

```

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_DrawCaptureImageBuffer(hScanner, hDC, nLeft, nTop, nRight, nBottom,
bCore);
if(ufs_res == UFS_OK) {
    // UFS_DrawCaptureImageBuffer is succeeded
} else {
    // UFS_DrawCaptureImageBuffer is failed
    // Use UFS\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nLeft As Long
Dim nTop As Long
Dim nRight As Long
Dim nBottom As Long
Dim bCore
' Get hScanner handleAs Long

' 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_STATUS.OK) Then
    ' UFS_DrawCaptureImageBuffer is succeeded
Else
    ' UFS_DrawCaptureImageBuffer is failed
    ' Use UFS\_GetErrorString function to show error string
End If

```

java (swing sample)

```
//load library(libScanner)
```

```
public void drawCurrentFingerImage()
{
/*test draw image*/
IntByReference refResolution = new IntByReference();
IntByReference refHeight   = new IntByReference();
IntByReference refWidth    = new IntByReference();
Pointer hScanner=null;

hScanner=GetCurrentScannerHandle();

libScanner.UFS_GetCaptureImageBufferInfo(hScanner, refWidth, refHeight,
refResolution);

byte[] pImageData = new byte[refWidth.getValue()*refHeight.getValue()];

libScanner.UFS_GetCaptureImageBuffer(hScanner,pImageData);

imgPanel.drawFingerImage(refWidth.getValue(),refHeight.getValue(),pImageData);
}

class ImagePanel extends JPanel
{
//private PlanarImage image;
private BufferedImage buffImage=null;

private void drawFingerImage(int nWidth,int nHeight,byte[] buff)
{
buffImage = new BufferedImage(nWidth, nHeight,
BufferedImage.TYPE_BYTE_GRAY);
buffImage.getRaster().setDataElements(0, 0, nWidth, nHeight, buff);

Graphics g = buffImage.createGraphics();
g.drawImage(buffImage, 0, 0, null);
g.dispose();
repaint();
}

public void paintComponent(Graphics g)
{
g.drawImage(buffImage, 0, 0, this);
}
}
```

UFS_DrawCaptureImageBuffer

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.



```
UFS_STATUS UFS_API UFS_DrawFeatureBuffer(
```

```
    HUFScanner hScanner,  
    HDC hDC,  
    int nLeft,  
    int nTop,  
    int nRight,  
    int nBottom,  
    int bCore  
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

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 or not

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_GetCaptureImageBufferInfo](#),
[UFS_GetCaptureImageBuffer](#), [UFS_SaveCaptureImageBufferToBMP](#),
[UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;
HUFScanner hScanner;
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_DrawFeatureBuffer is succeeded
} else {
    // UFS_DrawFeatureBuffer is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nLeft As Long
Dim nTop As Long
Dim nRight As Long
Dim nBottom As Long
Dim bCore
' Get hScanner handleAs Long
```

```
' 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_STATUS.OK) Then  
    'UFS_DrawFeatureBuffer is succeeded  
Else  
    'UFS_DrawFeatureBuffer is failed  
    'Use UFS_GetErrorString function to show error string  
End If
```

UFS_SaveCaptureImageBufferToBMP

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

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

*java JNA
int UFS_SaveCaptureImageBufferToBMP(
    Pointer hScanner,
    String szFileName
);

*java JNI
int UFS_SaveCaptureImageBufferToBMP(
    long hScanner,
    String szFileName
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_GetCaptureImageBufferInfo](#),
[UFS_GetCaptureImageBuffer](#), [UFS_DrawCaptureImageBuffer](#),
[UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; char szFileName[128]; // Get hScanner handle</pre>

```
// Get file name, szFileName

ufs_res = UFS_SaveCaptureImageBufferToBMP(hScanner, szFileName);
if(ufs_res == UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
} else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim FileName As String

' Get hScanner handle

' Get file name, FileName

ufs_res = UFS_SaveCaptureImageBufferToBMP(hScanner, FileName)
If(ufs_res = UFS_STATUS.OK) Then
    ' UFS_SaveCaptureImageBufferToBMP is succeeded
Else
    ' UFS_SaveCaptureImageBufferToBMP is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
String szFileName;

//load library(libScanner)

// Get hScanner handle

// Get file name, szFileName

ufs_res = libScanner.UFS_SaveCaptureImageBufferToBMP(hScanner, szFileName);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
} else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

// Get file name, szFileName

ufs_res = p.UFS_SaveCaptureImageBufferToBMP(hScanner[0], szFileName);
if(ufs_res == p.UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
} else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS\_GetErrorString function to show error string
}
```

UFS_SaveCaptureImageBufferTo19794_4

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

```
UFS_STATUS UFS_API UFS_SaveCaptureImageBufferTo19794_4(
    HUFScanner hScanner,
    char* szFileName
);

*java JNA
int UFS_SaveCaptureImageBufferTo19794_4(
    Pointer hScanner,
    String szFileName
);

*java JNI
int UFS_SaveCaptureImageBufferTo19794_4(
    long hScanner,
    String szFileName
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle to the scanner object
szFileName	[in] Specifies file name to save image buffer

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_GetCaptureImageBufferInfo](#),
[UFS_GetCaptureImageBuffer](#), [UFS_DrawCaptureImageBuffer](#),
[UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; char szFileName[128]; // Get hScanner handle</pre>

```
// Get file name, szFileName

ufs_res = UFS_SaveCaptureImageBufferTo19794_4(hScanner, szFileName);
if(ufs_res == UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
} else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim FileName As String

' Get hScanner handle

' Get file name, FileName

ufs_res = UFS_SaveCaptureImageBufferTo19794_4(hScanner, FileName)
If(ufs_res = UFS_STATUS.OK) Then
    ' UFS_SaveCaptureImageBufferToBMP is succeeded
Else
    ' UFS_SaveCaptureImageBufferToBMP is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
String szFileName;

//load library(libScanner)

// Get hScanner handle

// Get file name, szFileName

ufs_res = libScanner.UFS_SaveCaptureImageBufferTo19794_4(hScanner, szFileName);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
} else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

// Get file name, szFileName

ufs_res = p.UFS_SaveCaptureImageBufferTo19794_4(hScanner[0], szFileName);
if(ufs_res == p.UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
} else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS\_GetErrorString function to show error string
}
```

UFS_SaveCaptureImageBufferToWSQ

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

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

/*java JNA
int UFS_SaveCaptureImageBufferToWSQ(
    Pointer hScanner,
    float ratio,
    String szFileName
);

/*java JNI
int UFS_SaveCaptureImageBufferToWSQ(
    long hScanner,
    float ratio,
    String szFileName
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_GetCaptureImageBufferInfo](#),
[UFS_GetCaptureImageBuffer](#), [UFS_DrawCaptureImageBuffer](#),
[UFS_ClearCaptureImageBuffer](#)

Examples

Visual C++

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

// Get hScanner handle

// Get file name, szFileName

ufs_res = UFS_SaveCaptureImageBufferToWSQ(hScanner, ratio, szFileName);
if(ufs_res == UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
} else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim FileName As String

' Get hScanner handle

' Get file name, FileName

ufs_res = UFS_SaveCaptureImageBufferToWSQ(hScanner, ratio, FileName)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_SaveCaptureImageBufferToBMP is succeeded
Else
    ' UFS_SaveCaptureImageBufferToBMP is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;
String szFileName;

//load library(libScanner)

// Get hScanner handle

// Get file name, szFileName

ufs_res = libScanner.UFS_SaveCaptureImageBufferToWSQ(hScanner, ratio,
szFileName);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
```

```
    } else {
        // UFS_SaveCaptureImageBufferToBMP is failed
        // Use UFS\_GetErrorString function to show error string
    }
```

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

// Get file name, szFileName

ufs_res = p.UFS_SaveCaptureImageBufferToWSQ(hScanner[0], ratio, szFileName);
if(ufs_res == p.UFS_OK) {
    // UFS_SaveCaptureImageBufferToBMP is succeeded
} else {
    // UFS_SaveCaptureImageBufferToBMP is failed
    // Use UFS\_GetErrorString function to show error string
}
```

UFS_ClearCaptureImageBuffer

Clears the capture image buffer.

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

*java JNA
int UFS_ClearCaptureImageBuffer(Pointer hScanner);

*java JNI
int UFS_ClearCaptureImageBuffer(long hScanner);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

hScanner	[in] Handle to the scanner object
----------	-----------------------------------

See also

[UFS_CaptureSingleImage](#), [UFS_StartCapturing](#), [UFS_GetCaptureImageBufferInfo](#),
[UFS_GetCaptureImageBuffer](#), [UFS_DrawCaptureImageBuffer](#),
[UFS_SaveCaptureImageBufferToBMP](#)

Examples

Visual C++

```
UFS_STATUS ufs_res;
HUFScanner hScanner;

// Get hScanner handle

ufs_res = UFS_ClearCaptureImageBuffer(hScanner);
if(ufs_res == UFS_OK) {
    // UFS_ClearCaptureImageBuffer is succeeded
} else {
    // UFS_ClearCaptureImageBuffer is failed
    // Use UFS\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long

' Get hScanner handle

ufs_res = UFS_ClearCaptureImageBuffer(hScanner)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_ClearCaptureImageBuffer is succeeded
Else
    ' UFS_ClearCaptureImageBuffer is failed
    ' Use UFS\_GetErrorString function to show error string
End If
```

java JNA

```
int ufs_res;
Pointer hScanner;

//load library(libScanner)

// Get hScanner handle

ufs_res = libScanner.UFS_ClearCaptureImageBuffer(hScanner);
if(ufs_res == libScanner.UFS_OK) {
    // UFS_ClearCaptureImageBuffer is succeeded
} else {
    // UFS_ClearCaptureImageBuffer is failed
    // Use UFS\_GetErrorString function to show error string
}
```

java JNI

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

ufs_res = p.UFS_ClearCaptureImageBuffer(hScanner[0]);
if(ufs_res == p.UFS_OK) {
    // UFS_ClearCaptureImageBuffer is succeeded
} else {
    // UFS_ClearCaptureImageBuffer is failed
    // Use UFS\_GetErrorString function to show error string
}
```

UFS_GetErrorString

Gets the error string for specified [UFS_STAUS](#) value.

```
UFS\_STATUS UFS_API UFS_GetErrorString(
    UFS\_STATUS res,
    char* szErrorString
);

*java
int UFS_GetErrorString(
    int res,
    byte[] szErrorString
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

res	[in] Status return value
szErrorString	[out] Receives error sting

Examples

Visual C++

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

// Get status return code, ufs_res

ufs_res = UFS_GetErrorString(ufs_res, strError);
if(ufs_res == UFS_OK) {
    // UFS_GetErrorString is succeeded
} else {
    // UFS_GetErrorString is failed
}
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim m_strError As String

' Get status return code, ufs_res
```

```

ufs_res = UFS_GetErrorString(ufs_res, strError)
If (ufs_res = UFS_STATUS.OK) Then
    ' UFS_GetErrorString is succeeded
Else
    ' UFS_GetErrorString is failed
End If

```

```

java JNA
int ufs_res;
byte[] strError = new byte[128];

// Get status return code, ufs_res

ufs_res = libScanner.UFS_GetErrorString(ufs_res, strError);
if (ufs_res == libScanner.UFS_OK) {
    // UFS_GetErrorString is succeeded
} else {
    // UFS_GetErrorString is failed
}

```

```

java JNI
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
}

```

UFS_GetTemplateType

Gets parameter value.

```
UFS_STATUS UFS_API UFS_GetTemplateType(
    HUFScanner hScanner,
    void* pValue
);

*java JNA
int UFS_GetTemplateType(
    Pointer hScanner,
    IntByReference pValue
);

*java JNI
int UFS_SetTemplateType(
    long hScanner,
    int nTemplateType
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

<code>hScanner</code>	[in] Handle to the scanner object
<code>pValue</code>	[out] Receives parameter value of specified parameter type; <code>pValue</code> must point to adequate storage type matched to parameter type

See also

[UFS_SetTemplateType](#)

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; int nValue; // Get hScanner handle</pre>

```
ufs_res = UFS_GetTemplateType(hScanner,&nValue);
// Error handling routine is omitted
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nValue As Long

' Get hScanner handle

ufs_res = UFS_GetTemplateType(hScanner,nValue)
' Error handling routine is omitted
```

java JNA

```
int ufs_res;
Pointer hScanner;
int nTemplateType;
IntByReference refValue = new IntByReference();

//load library(libScanner)

// Get hScanner handle

ufs_res = libScanner.UFS_GetTemplateType(hScanner, refValue );
nTemplateType=refValue.getValue();
// Error handling routine is omitted
```

java JNI

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

ufs_res = p.UFS_GetTemplateType(hScanner[0], nTemplateType);
// Error handling routine is omitted
```

UFS_SetTemplateType

Gets parameter value.

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

*java JNA
int UFS_SetTemplateType(
    Pointer hScanner,
    int nTemplateType
);

*java JNI
int UFS_GetTemplateType(
    long hScanner,
    int[] nTemplateType
);
```

Possible return values

[UFS_OK](#), [UFS_ERROR](#), [UFS_ERR_INVALID_PARAMETERS](#)

Parameters

hScanner	[in] Handle to the scanner object
nTemplateType	[in] Parameter type; one of parameters

See also

[UFS_GetTemplateType](#)

Examples

Visual C++
<pre>UFS_STATUS ufs_res; HUFScanner hScanner; int nValue; // Get hScanner handle nValue = UFS_TEMPLATE_TYPE_SUPREMA;</pre>

```
ufs_res = UFS_SetTemplateType(hScanner,nValue);
// Error handling routine is omitted
```

Visual Basic 6.0

```
Dim ufs_res As UFS_STATUS
Dim hScanner As Long
Dim nValue As Long

' Get hScanner handle
nValue = UFS_TEMPLATE_TYPE_SUPREMA

ufs_res = UFS_SetTemplateType(hScanner,nValue)
' Error handling routine is omitted
```

java JNA

```
int ufs_res;
Pointer hScanner;
int nTemplateType;
int nValue;

//load library(libScanner)

// Get hScanner handle
nValue=UFS_TEMPLATE_TYPE_SUPREMA;

ufs_res = libScanner.UFS_SetTemplateType(hScanner, nValue);
// Error handling routine is omitted
```

java JNI

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

// make class library instance(BioMiniJniSDK p)

// Get hScanner handle

nValue=UFS_TEMPLATE_TYPE_SUPREMA;

ufs_res = p.UFS_SetTemplateType(hScanner[0], nValue);
// Error handling routine is omitted
```

UFS_SelectTemplate

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

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

*java JNA
int UFS_SelectTemplate_J(
    Pointer hScanner,
    PointerByReference ppTemplateInput,
    int[] pnTemplateInputSize,
    int nTemplateInputNum,
    PointerByReference ppTemplateOutput,
    int[] pnTemplateOutputSize,
    int nTemplateOutputNum)

*java JNI
int UFS_SelectTemplate(
    long hScanner,
    byte[][] ppTemplateInput,
    int[] pnTemplateInputSize,
    int nTemplateInputNum,
    byte[][] ppTemplateOutput,
    int[] pnTemplateOutputSize,
    int nTemplateOutputNum)
```

Possible return values

[UFS_OK](#), [UFS_ERR_INVALID_PARAMETERS](#), [UFS_ERR_NOT_SUPPORTED](#)

Parameters

hScanner	[in] Handle to the scanner object
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

See also[UFS_Extract](#)**Examples**

Visual C++
<pre>unsigned char Template[MAX_TEMPLATE_SIZE]; int TemplateSize; int nEnrollQuality; UFS_STATUS ufs_res; int i = 0; // 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); // UFS_SelectTemplate is succeed if(ufs_res == UFS_OK) { // If you want to enroll the output templates, move OutputTemplateArray and // OutputTemplateSizeArray data to your template array for enrollment or database. }</pre>

```
// select template function has error.  
else  
{  
    // ...  
}
```

UFMatcher module

UFMatcher module provides functionality for verifying fingerprints using two templates, identifying fingerprints using the template array, etc.

Requirements

Visual C++

- Required header: include\UFMatcher.h
- Required lib: lib\UFMatcher.lib
- Required dll: bin\UFMatcher.dll

Visual Basic 6.0

- Required reference: Suprema type library (bin\Suprema.tlb)
- Required dll: bin\UFMatcher.dll

Definitions

Status return value (UFM_STATUS)

Every function in UFMatcher module returns UFM_STATUS (integer) value having one of following values,

Status value definition	Code	Meaning
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 is not match
UFM_ERR_LICENSE_EXPIRED	-103	License is expired
UFM_ERR_NOT_SUPPORTED	-111	This function is not supported
UFM_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFM_ERR_MATCH_TIMEOUT	-401	Matching is timeout
UFM_ERR_MATCH_ABORTED	-402	Matching is aborted
UFM_ERR_TEMPLATE_TYPE	-411	Template type is not match

Template type

UFM_GetTemplateType(), UFM_SetTemplateType() functions use parameters defined as follows,

template type definition	Code	Meaning
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

Parameters

[UFM_GetParameter\(\)](#), [UFM_SetParameter\(\)](#) functions use parameters defined as follows,

Parameter value definition	Code	Meaning	Default value		
UFM_PARAM_FAST_MODE	301	Fast Mode (0: not use fast mode, 1: use fast mode)	1		
UFM_PARAM_SECURITY_LEVEL	302	<table border="1"> <tr> <td>Level</td> <td>False Accept Ratio (FAR)</td> </tr> </table>	Level	False Accept Ratio (FAR)	4
Level	False Accept Ratio (FAR)				

		<table border="1"> <tr><td>1</td><td>Below 1 % (1e-2)</td></tr> <tr><td>2</td><td>Below 0.1 % (1e-3)</td></tr> <tr><td>3</td><td>Below 0.01 % (1e-4)</td></tr> <tr><td>4</td><td>Below 0.001 % (1e-5)</td></tr> <tr><td>5</td><td>Below 0.0001 % (1e-6)</td></tr> <tr><td>6</td><td>Below 0.00001 % (1e-7)</td></tr> <tr><td>7</td><td>Below 0.000001 % (1e-8)</td></tr> </table>	1	Below 1 % (1e-2)	2	Below 0.1 % (1e-3)	3	Below 0.01 % (1e-4)	4	Below 0.001 % (1e-5)	5	Below 0.0001 % (1e-6)	6	Below 0.00001 % (1e-7)	7	Below 0.000001 % (1e-8)	
1	Below 1 % (1e-2)																
2	Below 0.1 % (1e-3)																
3	Below 0.01 % (1e-4)																
4	Below 0.001 % (1e-5)																
5	Below 0.0001 % (1e-6)																
6	Below 0.00001 % (1e-7)																
7	Below 0.000001 % (1e-8)																
UFM_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0														

Matcher handle

HUFMatcher defines handle to UFMatcher object.

```
typedef void* HUFMatcher;
```

*java JNA pointer type defines handle to UFMatcherClass

```
public Pointer hMaatcher=null;
```

UFM_Create

Creates a matcher.

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

*java
int UFM_Create(PointerByReference phMatcher);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_NO_LICENSE](#),
[UFM_ERR_LICENSE_NOT_MATCH](#), [UFM_ERR_LICENSE_EXPIRED](#),
[UFM_ERR_INVALID_PARAMETERS](#)

Parameters

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

See also

[UFM_Delete](#)

Examples

Visual C++

```
UFM_STATUS ufm_res;
HUFSMatcher hMatcher;

ufm_res = UFM_Create(&hMatcher);
if(ufm_res == UFM_OK) {
    // UFM_Create is succeeded
} else {
    // UFM_Create is failed
    // Use UFM\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufm_res As UFM_STATUS
Dim hMatcher As Long

ufm_res = UFM_Create(hMatcher)
```

```
If(ufm_res == UFM_STATUS.OK) Then
    ' UFM_Create is succeeded
Else
    ' UFM_Create is failed
    ' Use UFM\_GetErrorString function to show error string
End If
```

```
java JNA
int ufm_res;
Pointer hMatcher=null;
PointerByReference refMatcher = new PointerByReference();
UFMatcherClass libMatcher=null;

//load library(libMatcher)
libMatcher = (UFMatcherClass)Native.loadLibrary("UFMatcher",UFMatcherClass.class);

ufm_res = libMatcher.UFM_Create(refMatcher);
if(ufm_res == libMatcher.UFM_OK) {
    // UFM_Create is succeeded
    hMatcher = refMatcher.getValue();
} else {
    // UFM_Create is failed
    // Use UFM\_GetErrorString function to show error string
}
```

```
java JNI
int ufm_res;
long[] hMatcher = new long[1];

//make instance
p = new BioMiniJniSDK();

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
}
```

UFM_Delete

Deletes specified matcher.

```
UFM_STATUS UFM_API UFM_Delete(
    HUFMatcher pMatcher
);

*java JNA
int UFM_Delete(Pointer hMatcher);

*java JNI
int UFM_Delete(long hMatcher);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_INVALID_PARAMETERS](#)

Parameters

hMatcher	[in] Handle of the matcher object
----------	-----------------------------------

See also

[UFM_Create](#)

Examples

Visual C++

```
UFM_STATUS ufm_res;
HUFMatcher hMatcher;

// Create hMatcher and use

ufm_res = UFM_Delete(hMatcher);
if(ufm_res == UFM_OK) {
    // UFM_Delete is succeeded
} else {
    // UFM_Delete is failed
    // Use UFM\_GetErrorString function to show error string
}
```

Visual Basic 6.0

Dim ufm_res As UFM_STATUS

```

Dim hMatcher As Long

' Create hMatcher and use

ufm_res = UFM_Delete(hMatcher)
If (ufm_res = UFM_STATUS.OK) Then
    ' UFM_Delete is succeeded
Else
    ' UFM_Delete is failed
    ' Use UFM\_GetErrorString function to show error string
End If

```

```

java JNA
int ufm_res;
Pointer hMatcher;

//load library(libMatcher)

// Create hMatcher and use

ufm_res = libMatcher.UMF_Delete(hMatcher);
if (ufm_res == libMatcher.UMF_OK) {
    // UFM_Delete is succeeded
} else {
    // UFM_Delete is failed
    // Use UFM\_GetErrorString function to show error string
}

```

```

java JNI
int ufm_res;
long[] hMatcher = new long[1];

// make class library instance(BioMiniJniSDK p)

// Create hMatcher handle

ufm_res = p.UMF_Delete(hMatcher[0]);
if (ufm_res == p.UMF_OK) {
    // UFM_Delete is succeeded
} else {
    // UFM_Delete is failed
    // Use UFM\_GetErrorString function to show error string
}

```

UFM_GetParameter

Gets parameter value.

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

*java JNA
int UFM_GetParameter(
    Pointer hMatcher,
    int nParam,
    IntByReference pValue
);

*java JNI
int UFM_GetParameter(
    long hMatcher,
    int nParam,
    int[] pValue
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_INVALID_PARAMETERS](#)

Parameters

<u>hMatcher</u>	[in] Handle of the matcher object
<u>nParam</u>	[in] Parameter type; one of parameters
<u>pValue</u>	[out] Receives parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

See also

[UFM_SetParameter](#)

Examples

Visual C++
UFM_STATUS ufm_res; HUMatcher hMatcher;

```

int nValue;

// Create hMatcher

// Get fast mode
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_FAST_MODE, &nValue);
// Error handling routine is omitted

// Get security level
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL,
&nValue);
// Error handling routine is omitted

// Get use SIF
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted

```

Visual Basic 6.0

```

Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim nValue As Long

' Create hMatcher

' Get fast mode
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_FAST_MODE, nValue)
' Error handling routine is omitted

' Get security level
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL, nValue)
' Error handling routine is omitted

' Get use SIF
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_USE_SIF, nValue)
' Error handling routine is omitted

```

java JNA

```

int ufm_res;
Pointer hMatcher;
IntByReference nValue = new IntByReference();

//load library(libMatcher)

// Create hMatcher

// Get fast mode
ufm_res = libMatcher.UEM_GetParameter(hMatcher, UEM_PARAM_FAST_MODE,
nValue);

```

```
// Error handling routine is omitted

// Get security level
ufm_res = libMatcher.UMF_GetParameter(hMatcher,
UFM_PARAM_SECURITY_LEVEL, nValue);
// Error handling routine is omitted

// Get use SIF
ufm_res = UFM_GetParameter(hMatcher, UFM_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted
```

```
java JNI
int ufm_res;
long[] hMatcher = new long[1];

// make class library instance(BioMiniJniSDK p)

// Create hMatcher handle

// Get fast mode
ufm_res = p.UMF_GetParameter(hMatcher[0], p.UMF_PARAM_FAST_MODE,
nValue);
// Error handling routine is omitted

// Get security level
ufm_res = p.UMF_GetParameter(hMatcher[0], p.UMF_PARAM_SECURITY_LEVEL,
nValue);
// Error handling routine is omitted

// Get use SIF
ufm_res = p.UMF_GetParameter(hMatcher[0], p.UMF_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted
```

UFM_SetParameter

Sets parameter value.

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

*java JNA
int UFM_SetParameter(
    Pointer hMatcher,
    int nParam,
    IntByReference pValue
);

*java JNI
int UFM_SetParameter(
    long hMatcher,
    int nParam,
    int[] pValue
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_INVALID_PARAMETERS](#)

Parameters

hMatcher	[in] Handle of the matcher object
nParam	[in] Parameter type; one of parameters
pValue	[in] Pointer to parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

See also

[UFM_GetParameter](#)

Examples

Visual C++
UFM_STATUS ufm_res; HUFMatcher hMatcher;

```
int nValue;

// Create hMatcher

// Set fast mode to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_FAST_MODE, &nValue);
// Error handling routine is omitted

// Set security level to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL,
&nValue);
// Error handling routine is omitted

// Set use SIF to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted
```

Visual Basic 6.0

```
Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim nValue As Long

' Create hMatcher

' Set fast mode to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_FAST_MODE, nValue)
' Error handling routine is omitted

' Set security level to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL, nValue)
' Error handling routine is omitted

' Set use SIF to nValue
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_USE_SIF, nValue)
' Error handling routine is omitted
```

java JNA

```
int_STATUS ufm_res;
Pointer hMatcher;
IntByReference nValue = new IntByReference();

//load library(libMatcher)

// Create hMatcher

// Set fast mode to nValue
nValue.setValue(1);
ufm_res = libMatcher.UMF_SetParameter(hMatcher, UFM_PARAM_FAST_MODE,
```

```

&nValue);
// Error handling routine is omitted

// Set security level to nValue
nValue.setValue(4);
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_SECURITY_LEVEL,
&nValue);
// Error handling routine is omitted

// Set use SIF to nValue
nValue.setValue(0);
ufm_res = UFM_SetParameter(hMatcher, UFM_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted

```

```

java JNI
int_STATUS ufm_res;
long[] hMatcher = new long[1];
int[] nValue = new int[1];

// make class library instance(BioMiniJniSDK p)

// Create hMatcher handle

// Set fast mode to nValue
nValue[0] = 1;
ufm_res = p.UMF_SetParameter(hMatcher[0], p.UMF_PARAM_FAST_MODE, nValue);
// Error handling routine is omitted

// Set security level to nValue
nValue[0] = 4;
ufm_res = UFM_SetParameter(hMatcher[0], p.UMF_PARAM_SECURITY_LEVEL,
nValue);
// Error handling routine is omitted

// Set use SIF to nValue
nValue[0] = 0;
ufm_res = UFM_SetParameter(hMatcher[0], p.UMF_PARAM_USE_SIF, nValue);
// Error handling routine is omitted

```

UFM_Verify

Compares two extracted templates.

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

*java JNA
int UFM_Verify(
    Pointer hMatcher,
    byte[] pTemplate1,
    int nTemplate1Size,
    byte[] pTemplate2,
    int nTemplate2Size,
    IntByReference bVerifySucceed
);

*java JNI
int UFM_Verify(
    long hMatcher,
    byte[] pTemplate1,
    int nTemplate1Size,
    byte[] pTemplate2,
    int nTemplate2Size,
    int[] bVerifySucceed
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_LICENSE_NOT_MATCH](#),
[UFM_ERR_LICENSE_EXPIRED](#), [UFM_ERR_INVALID_PARAMETERS](#),
[UFM_ERR_TEMPLATE_TYPE](#)

Parameters

hMatcher	[in] Handle of the matcher object
pTemplate1	[in] Pointer to the template
nTemplate1Size	[in] Specifies the size of the template

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

Examples

Visual C++

```

// Assume template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384

UFM_STATUS ufm_res;
HUFMatcher hMatcher;
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 is succeeded
    if(bVerifySucceed) {
        // 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
}

```

Visual Basic 6.0

```

' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2Size As Long
Dim VerifySucceed As Long

```

```

' Create hMatcher

' Get two templates, Template1 and Template2

ufm_res = UFM_Verify(hMatcher, Template1(0), Template1Size, Template2(0),
Tempalte2Size, VerifySucceed)
If (ufm_res = UFM_STATUS.OK) Then
    ' UFM_Verify is succeeded
    If (VerifySucceed = 1) Then
        ' Template1 is matched to Template2
    Else
        ' Template1 is not matched to Template2
    End If
Else
    ' UFM_Verify is failed
    ' Use UFM\_GetErrorString function to show error string
End If

```

```

java JNA
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufm_res;
Pointer hMatcher;
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int nTemplate1Size;
int nTemplate2Size;
IntByReference bVerifySucceed = new IntByReference\(\);
int nSucceed;

//load library(libMatcher)

// Create hMatcher

// Get two templates, Template1 and Template2

ufm_res = libMatcher.UFM_Verify(hMatcher, Template1, nTemplate1Size, Template2,
nTemplate2Size, bVerifySucceed);
if (ufm_res == libMatcher.UMF_OK) {
    // UFM_Verify is succeeded
    nSucceed=bVerifySucceed.getValue();
    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
}
```

```
java JNI
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

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(BioMiniJniSDK p)

// Create hMatcher handle

// Get two templates, Template1 and Template2

ufm_res = p.UMF_Verify(hMatcher[0], Template1, nTemplate1Size, Template2,
nTemplate2Size, bVerifySucceed);
if(ufm_res == p.UMF_OK) {
    // UFM_Verify is succeeded
    nSucceed=bVerifySucceed[0];
    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
}
```

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.

```
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
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_LICENSE_NOT_MATCH](#),
[UFM_ERR_LICENSE_EXPIRED](#), [UFM_ERR_INVALID_PARAMETERS](#),
[UFM_ERR_MATCH_TIMEOUT](#), [UFM_ERR_MATCH_ABORTED](#),
[UFM_ERR_TEMPLATE_TYPE](#)

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

See also[UFM_AbortIdentify](#)**Examples**

Visual C++
<pre>// Assume template size is 384 bytes #define MAX_TEMPLATE_SIZE 384 UFM_STATUS ufm_res; HUFMatcher hMatcher; 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, Template1 // 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 = 0; 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) {</pre>

```

// UFM_Identify is succeeded
if(nMatchTemplate2Index != -1) {
    // Input fingerprint Template1 is matched to
    ppTemplate2[nMatchTemplate2Index]
} else {
    // 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 = 0; i < nTemplate2Num; i++) {
    free(ppTemplate2[i]);
}
free(ppTemplate2);

```

Visual Basic 6.0

```

' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2() As Byte
Dim Template2Size() As Long
Dim Template2Ptr() As Long
Dim Template2Num As Long
Dim Timeout As Long
Dim MatchTemplate2Index As Long

' Create hMatcher

' Get input template from user, Template1

' Make template array from DB or something
' Get number of template to nTemplate2Num
ReDim Template2(MAX_TEMPLATE_SIZE - 1, Template2Num - 1) As Byte
ReDim Template2Size(Template2Num - 1) As Long

' Copy i th template to Template2(i)
' Set i th template size to Template2Size(i)

' Make template pointer array to pass two dimensional template data
ReDim Template2Ptr(Template2Num - 1) As Long
For i = 0 To Template2Num - 1

```

```
Template2Ptr(i) = VarPtr(Template2(0, i))
```

Next

```
ufm_res = UFM_Identify(hMatcher, Template1(0), Template1Size, Template2Ptr(0),  
Template2Size(0), nTemplate2Num, Timeout, MatchTemplate2Index)
```

```
If (ufm_res = UFM_STATUS.OK) Then
```

```
    ' UFM_Identify is succeeded
```

```
    If (MatchTemplate2Index <> -1) Then
```

```
        ' Input fingerprint Template1 is matched to  
        Template2(MatchTemplate2Index)
```

```
    Else
```

```
        ' Input fingerprint is not in Template2
```

```
    End If
```

```
Else
```

```
    ' UFM_Identify is failed
```

```
    ' Use UFM\_GetErrorString function to show error string
```

```
End If
```

UFM_AbortIdentify

Aborts current identifying procedure started using [UFM_Identify\(\)](#).

```
UFM\_STATUS UFM_API UFM_AbortIdentify(
    HUFMatcher pMatcher
);

*java JNA
int UFM_AbortIdentify(Pointer hMatcher);

*java JNI
int UFM_AbortIdentify(long hMatcher);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_INVALID_PARAMETERS](#)

Parameters

hMatcher	[in] Handle of the matcher object
----------	-----------------------------------

See Also

[UFM_Identify](#)

Examples

Visual C++

```
UFM_STATUS ufm_res;
HUFMatcher hMatcher;

// Create hMatcher

// Start UFM_Identify

ufm_res = UFM_AbortIdentify(hMatcher);
if(ufm_res == UFM_OK) {
    // UFM_AbortIdentify is succeeded
} else {
    // UFM_AbortIdentify is failed
    // Use UFM\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufm_res As UFM_STATUS
Dim hMatcher As Long

' Create hMatcher

' Start UFM_Identify

ufm_res = UFM_AbortIdentify(hMatcher)
If (ufm_res = UFM_STATUS.OK) Then
    ' UFM_AbortIdentify is succeeded
Else
    ' UFM_AbortIdentify is failed
    ' Use UFM\_GetErrorString function to show error string
End If
```

java JNA

```
int ufm_res;
Pointer hMatcher;

//load library(libMatcher)

// Create hMatcher

// Start UFM_Identify

ufm_res = libMatcher.UFM_AbortIdentify(hMatcher);
if (ufm_res == libMatcher.UMF_OK) {
    // UFM_AbortIdentify is succeeded
} else {
    // UFM_AbortIdentify is failed
    // Use UFM\_GetErrorString function to show error string
}
```

java JNI

```
int ufm_res;
long[] hMatcher = new long[1];

// make class library instance(BioMiniJniSDK p)

// Create hMatcher handle

// Start UFM_Identify

ufm_res = p.UFM_AbortIdentify(hMatcher);
if (ufm_res == p.UMF_OK) {
    // UFM_AbortIdentify is succeeded
} else {
```

```
// UFM_AbortIdentify is failed  
// Use UFM\_GetErrorString function to show error string
```

UFM_IdentifyInit

Initializes identify with input template.

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

*java JNA
int UFM_IdentifyInit(
    Pointer hMatcher,
    byte[] pTemplate1,
    int nTemplate1Size
);

*java JNI
int UFM_IdentifyInit(
    long hMatcher,
    byte[] pTemplate1,
    int nTemplate1Size
);
```

Possible return values

UFM_OK, UFM_ERROR, UFM_ERR_LICENSE_NOT_MATCH,
UFM_ERR_LICENSE_EXPIRED, UFM_ERR_INVALID_PARAMETERS

Parameters

hMatcher	[in] Handle of the matcher object
pTemplate1	[in] Pointer to the template
nTemplate1Size	[in] Specifies the size of the template

See also

[UFM_IdentifyNext](#)

Examples

Visual C++
// Assume template size is 384 bytes

```
#define MAX_TEMPLATE_SIZE 384

UFM_STATUS ufm_res;
HUFMatcher hMatcher;
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 is succeeded
} else {
    // UFM_IdentifyInit is failed
    // Use UFM\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long

' Create hMatcher

' Get Template1

ufm_res = UFM_IdentifyInit(hMatcher, Template1(0), Template1Size)
If(ufm_res = UFM_STATUS.OK) Then
    ' UFM_IdentifyInit is succeeded
Else
    ' UFM_IdentifyInit is failed
    ' Use UFM\_GetErrorString function to show error string
End If
```

java JNA

```
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufm_res;
Pointer hMatcher;
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
int nTemplate1Size;
```

```
//load library(libMatcher)

// Create hMatcher

// Get Template1

ufm_res = libMatcher.UMF_IdentifyInit(hMatcher, Template1, nTemplate1Size);
if(ufm_res == libMatcher.UMF_OK) {
    // UMF_IdentifyInit is succeeded
} else {
    // UMF_IdentifyInit is failed
    // Use UMF\_GetErrorString function to show error string
}
```

```
java JNI
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufm_res;
long[] hMatcher = new long[1];
byte[] Template1 = new byte[MAX_TEMPLATE_SIZE];
int nTemplate1Size;

// make class library instance(BioMiniJniSDK p)

// Create hMatcher handle

// Get Template1

ufm_res = p.UMF_IdentifyInit(hMatcher[0], Template1, nTemplate1Size);
if(ufm_res == p.UMF_OK) {
    // UMF_IdentifyInit is succeeded
} else {
    // UMF_IdentifyInit is failed
    // Use UMF\_GetErrorString function to show error string
}
```

UFM_IdentifyNext

Matches one input template to the template specified in [UFM_IdentifyInit\(\)](#).

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

*java JNA
int UFM_IdentifyNext(
    Pointer hMatcher,
    byte[] pTemplate2,
    int nTemplate2Size,
    IntByReference bIdentifySucceed
);

*java JNI
int UFM_IdentifyNext(
    long hMatcher,
    byte[] pTemplate2,
    int nTemplate2Size,
    int[] bIdentifySucceed
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_LICENSE_NOT_MATCH](#),
[UFM_ERR_LICENSE_EXPIRED](#), [UFM_ERR_INVALID_PARAMETERS](#),
[UFM_ERR_TEMPLATE_TYPE](#)

Parameters

hMatcher	[in] Handle of 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

See also

[UFM_IdentifyInit](#)

Examples

Visual C++

```
// Assume template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384

UFM_STATUS ufm_res;
HUFMatcher hMatcher;
unsigned char Template2[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
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 = 0;
for (i = 0; i < nTemplate2Num; i++) {
    // Get one template in DB or something, and save it to Template2 and
    // nTemplate2Size

    ufm_res = UFM_IdentifyNext(hMatcher, Template2, nTemplate2Size,
        bIdentifySucceed);
    if(ufm_res == 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
}
```

Visual Basic 6.0

```
' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384
```

```

Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2Size As Long
Dim Template2Num As Long
Dim IdentifySucceed As Long
Dim i As Long

' Create hMatcher

' Execute UFM_IdentifyInit with query template

' Get number of templates in DB or something, and save it to Template2Num

IdentifySucceed = 0
for i = 0 To Template2Num - 1
    ' Get one template in DB or something, and save it to Template2 and Template2Size

    ufm_res = UFM_IdentifyNext(hMatcher, Template2, Template2Size,
                                IdentifySucceed)
    If (ufm_res = UFM_STATUS.OK) Then
        ' UFM_IdentifyNext is succeeded
    Else
        ' UFM_IdentifyNext is failed
        ' Use UFM\_GetErrorString function to show error string
        ' Exit
    End If

    If (bIdentifySucceed = 1) Then
        ' Identification is succeed
        Exit For
    End If
Next
If (bIdentifySucceed = 0) Then
    ' Identification is failed
End If

```

```

java JNA
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufm_res;
Pointer hMatcher;
byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
int nTemplate2Num;
IntByReference bIdentifySucceed = new IntByReference();
int i;

```

```

//load library(libMatcher)

// Create hMatcher

// Execute UFM_IdentifyInit with query template

// Get number of templates in DB or something, and save it to nTemplate2Num

for (i = 0; i < nTemplate2Num; i++) {
    // Get one template in DB or something, and save it to Template2 and
    nTemplate2Size

    ufm_res = libMatcher.UMF_IdentifyNext(hMatcher, Template2, nTemplate2Size,
    bIdentifySucceed);
    if(ufm_res == libMatcher.UMF_OK) {
        // UFM_IdentifyNext is succeeded
    } else {
        // UFM_IdentifyNext is failed
        // Use UMF\_GetErrorString function to show error string
        // return;
    }

    if(bIdentifySucceed.getValue()) {
        // Identification is succeed
        break;
    }
}
if(!bIdentifySucceed.getValue()) {
    // Identification is failed
}

```

java JNI
<pre> // Assume template size is 384 bytes int MAX_TEMPLATE_SIZE = 384; int ufm_res; long[] hMatcher = new long[1]; byte[] Template2 = new byte[MAX_TEMPLATE_SIZE]; int nTemplate2Size; int nTemplate2Num; int[] bIdentifySucceed = new int[1]; int i; // make class library instance(BioMiniJniSDK p) // Create hMatcher handle // Get number of templates in DB or something, and save it to nTemplate2Num </pre>

```
for (i = 0; i < nTemplate2Num; i++) {
    // Get one template in DB or something, and save it to Template2 and
    nTemplate2Size

    ufm_res = p.UMF_IdentifyNext(hMatcher[0], Template2, nTemplate2Size,
        bIdentifySucceed);
    if(ufm_res == p.UMF_OK) {
        // UFM_IdentifyNext is succeeded
    } else {
        // UFM_IdentifyNext is failed
        // Use UMF_GetErrorString function to show error string
        // return;
    }

    if(bIdentifySucceed[0]) {
        // Identification is succeed
        break;
    }
}
if(!bIdentifySucceed[0]) {
    // Identification is failed
}
```

UFM_RotateTemplate

Rotates the specified template to the amount of 180 degrees.

```
UFM\_STATUS UFM_API UFM_RotateTemplate(
    HUFMatcher pMatcher,
    unsigned char* pTemplate,
    int nTemplateSize
);

*java JNA
int UFM_RotateTemplate(
    Pointer hMatcher,
    byte[] pTemplate,
    int nTemplateSize
);

*java JNI
int UFM_RotateTemplate(
    long hMatcher,
    byte[] pTemplate,
    int nTemplateSize
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_INVALID_PARAMETERS](#)

Parameters

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

Examples

```
Visual C++
// Assume template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384

UFM_STATUS ufm_res;
HUFMatcher hMatcher;
unsigned char Template2[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 is succeeded
} else {
    // UFM_RotateTemplate is failed
    // Use UFM\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateSize As Long

' Create hMatcher

' Get a template, and save it to Template and TemplateSize

ufm_res = UFM_RotateTemplate(hMatcher, Template(0), TemplateSize)
If (ufm_res = UFM_STATUS.OK) Then
    ' UFM_RotateTemplate is succeeded
Else
    ' UFM_RotateTemplate is failed
    ' Use UFM\_GetErrorString function to show error string
End If
```

java JNA

```
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufm_res;
Pointer hMatcher;
byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int nTemplateSize;

//load library(libMatcher)

// Create hMatcher

// Get a template, and save it to Template and nTemplateSize
```

```
ufm_res = libMatcher.UMF_RotateTemplate(hMatcher, Template, nTemplateSize);
if(ufm_res == libMatcher.UMF_OK) {
    // UFM_RotateTemplate is succeeded
} else {
    // UFM_RotateTemplate is failed
    // Use UFM\_GetErrorString function to show error string
}
```

java JNI

```
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufm_res;
long[] hMatcher = new long[1];
byte[] Template2 = new byte[MAX_TEMPLATE_SIZE];
int nTemplateSize;

// make class library instance(BioMiniJniSDK p)

// Create hMatcher handle

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

ufm_res = p.UMF_RotateTemplate(hMatcher[0], Template, nTemplateSize);
if(ufm_res == p.UMF_OK) {
    // UFM_RotateTemplate is succeeded
} else {
    // UFM_RotateTemplate is failed
    // Use UFM\_GetErrorString function to show error string
}
```

UFM_GetErrorString

Gets the error string for specified [UFM_STAUS](#) value.

```
UFM\_STATUS UFM_API UFM_GetErrorString(
    UFM\_STATUS res,
    char* szErrorString
);

*java
int UFM_GetErrorString(
    int res,
    byte[] szErrorString
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_INVALID_PARAMETERS](#)

Parameters

res	[in] Status return value
szErrorString	[out] Receives error sting

Examples

Visual C++

```
UFM_STATUS ufm_res;
char strError[128];

// Get status return code, ufm_res

ufm_res = UFM_GetErrorString(ufm_res, strError);
if(ufm_res == UFM_OK) {
    // UFM_GetErrorString is succeeded
} else {
    // UFM_GetErrorString is failed
}
```

Visual Basic 6.0

```
Dim ufm_res As UFM_STATUS
Dim m_strError As String
```

```
' Get status return code, ufm_res

ufm_res = UFM_GetErrorString(ufm_res, strError)
If (ufm_res = UFM_STATUS.OK) Then
    ' UFM_GetErrorString is succeeded
Else
    ' UFM_GetErrorString is failed
End If
```

```
java JNA
int ufs_res;
byte[] strError = new byte[128];

// Get status return code, ufm_res

ufs_res = libMatcher.UMF_GetErrorString(ufs_res, strError);
if (ufs_res == libMatcher.UFS_OK) {
    // UFM_GetErrorString is succeeded
} else {
    // UFM_GetErrorString is failed
}
```

```
java JNI
int ufs_res;
byte[] strError = new byte[128];

// Get status return code, ufm_res

ufs_res = p.UMF_GetErrorString(ufs_res, strError);
if (ufs_res == p.UFS_OK) {
    // UFM_GetErrorString is succeeded
} else {
    // UFM_GetErrorString is failed
}
```

UFM_GetTemplateType

Gets parameter value.

```
UFM\_STATUS UFM_API UFM_GetTemplateType(
    HUFMatcher hMatcher,
    void\* pValue
);

*java JNA
int UFM_GetTemplateType(
    Pointer hMatcher,
    IntByReference pValue
);

*java JNI
int UFM_SetTemplateType(
    long hMatcher,
    int nTemplateType
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_INVALID_PARAMETERS](#)

Parameters

<code>hMatcher</code>	[in] Handle to the matcher object
<code>pValue</code>	[out] Receives parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

See also

[UFM_SetTemplateType](#)

Examples

Visual C++

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

// Get hMatcher handle
```

```
ufm_res = UFM_GetTemplateType(hMatcher,&nValue);
// Error handling routine is omitted
```

Visual Basic 6.0

```
Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim nValue As Long

' Get hMatcher handle

ufm_res = UFM_GetTemplateType(hMatcher,nValue)
' Error handling routine is omitted
```

java JNA

```
int ufm_res;
Pointer hMatcher;
int nTemplateType;
IntByReference refValue = new IntByReference();

//load library(libMatcher)

// Get hMatcher handle

ufm_res = libMatcher.UMF_GetTemplateType(hMatcher, refValue );
nTemplateType=refValue.getValue();
// Error handling routine is omitted
```

java JNI

```
int ufm_res;
long[] hMatcher = new long[1];
int[] nTemplateType = new int[1];

// make class library instance(BioMiniJniSDK p)

// Create hMatcher handle

ufm_res = p.UMF_GetTemplateType(hMatcher[0], nTemplateType );
// Error handling routine is omitted
```

UFM_SetTemplateType

Gets parameter value.

```
UFM\_STATUS UFM_API UFM_SetTemplateType(
    HUFMatcher hMatcher,
    int nTemplateType
);

*java JNA
int UFM_SetTemplateType(
    Pointer hMatcher,
    int nTemplateType
);

*java JNI
int UFM_GetTemplateType(
    long hMatcher,
    int\[\] nTemplateType
);
```

Possible return values

[UFM_OK](#), [UFM_ERROR](#), [UFM_ERR_INVALID_PARAMETERS](#)

Parameters

hMatcher	[in] Handle to the matcher object
nTemplateType	[in] Parameter type; one of parameters

See also

[UFM_GetTemplateType](#)

Examples

Visual C++
<pre>UFM_STATUS ufm_res; HUFMatcher hMatcher; int nValue; // Get hScanner handle nValue = UFM_TEMPLATE_TYPE_SUPREMA;</pre>

```
ufs_res = UFM_SetTemplateType(hMatcher,nValue);
// Error handling routine is omitted
```

Visual Basic 6.0

```
Dim ufm_res As UFM_STATUS
Dim hMatcher As Long
Dim nValue As Long

' Get hScanner handle
nValue = UFM_TEMPLATE_TYPE_SUPREMA

ufm_res = UFM_SetTemplateType(hMatcher,nValue)
' Error handling routine is omitted
```

java JNA

```
int ufm_res;
Pointer hMatcher;
int nTemplateType;
int nValue;

//load library(libMatcher)

// Get hMatcher handle
nValue=UFM_TEMPLATE_TYPE_SUPREMA;

ufm_res = libMatcher.UMF_SetTemplateType(hMatcher, nValue);
// Error handling routine is omitted
```

java JNI

```
int ufm_res;
long[] hMatcher = new long[1];
int nValue;

// make class library instance(BioMiniJniSDK p)

// Create hMatcher handle
nValue=p.UMF_TEMPLATE_TYPE_SUPREMA;

ufm_res = p.UMF_SetTemplateType(hMatcher[0], nValue);
// Error handling routine is omitted
```

UFExtractor module

UFExtractor module provides functionality for extracting templates from input images, etc.

Requirements

Visual C++

- Required header: include\UFExtractor.h
- Required lib: lib\UFExtractor.lib
- Required dll: bin\UFExtractor.dll

Visual Basic 6.0

- Required reference: Suprema type library (bin\Suprema.tlb)
- Required dll: bin\UFExtractor.dll

Definitions

Status return value (UFE_STATUS)

Every function in UFExtractor module returns UFE_STATUS (integer) value having one of following values,

Status value definition	Code	Meaning
UFE_OK	0	Success
UFE_ERROR	-1	General error
UFE_ERR_NO_LICENSE	-101	System has no license
UFE_ERR_LICENSE_NOT_MATCH	-102	License is not match
UFE_ERR_LICENSE_EXPIRED	-103	License is expired
UFE_ERR_NOT_SUPPORTED	-111	This function is not supported
UFE_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFE_ERR_NOT_GOOD_IMAGE	-301	Input image is not good
UFE_ERR_EXTRACTION_FAILED	-302	Extraction is failed
UFE_ERR_UNDEFINED_MODE	-311	Undefined mode
UFE_ERR_CORE_NOT_DETECTED	-351	Core is not detected
UFE_ERR_CORE_TO_LEFT	-352	Move finger to left
UFE_ERR_CORE_TO_LEFT_TOP	-353	Move finger to left-top
UFE_ERR_CORE_TO_TOP	-354	Move finger to top
UFE_ERR_CORE_TO_RIGHT_TOP	-355	Move finger to right-top
UFE_ERR_CORE_TO_RIGHT	-356	Move finger to right
UFE_ERR_CORE_TO_RIGHT_BOTTOM	-357	Move finger to right-bottom
UFE_ERR_CORE_TO_BOTTOM	-358	Move finger to bottom
UFE_ERR_CORE_TO_LEFT_BOTTOM	-359	Move finger to left-bottom

Template type

UFE_GetTemplateType(), UFE_SetTemplateType() functions use parameters defined as follows,

template type definition	Code	Meaning
UFE_TEMPLATE_TYPE_SUPREMA	2001	suprema template type
UFE_TEMPLATE_TYPE_ISO19794_2	2002	ISO template type
UFE_TEMPLATE_TYPE_ANSI378	2003	ANSI378 template type

Parameters

[UFE_GetParameter\(\)](#), [UFE_SetParameter\(\)](#) functions use parameters defined as follows,

Parameter value definition	Code	Meaning	Default value
UFE_PARAM_DETECT_CORE	301	Detect core (0: not use core, 1: use core)	0
UFE_PARAM_TEMPLATE_SIZE	302	Template size (256 ~ 1024, 32 bytes step size)	384
UFE_PARAM_USE_SIF	311	Use SIF (0: not use SIF, 1: use SIF)	0

Mode

[UFE_GetMode\(\)](#), [UFE_SetMode\(\)](#) functions use parameters defined as follows,

Parameter value definition	Code	Meaning
UFE_MODE_SFR200	1001	Suprema SFR200
UFE_MODE_SFR300	1002	Suprema SFR300-S
UFE_MODE_SFR300v2	1003	Suprema SFR300-S(Ver.2)
UFE_MODE_GENERAL	1004	General scanner type

Extractor handle

HUFExtractor defines handle to UFExtractor object.

```
typedef void* HUFExtractor;
```

*java JNA pointer type defines handle to UFExtractorClass

```
public Pointer hExtractor=null;
```

UFE_Create

Creates an extractor.

```
UFE_STATUS UFE_API UFE_Create(
    HUFExtractor* phExtractor
);

*java
int UFE_Create(PointerByReference hExtractor);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_NO_LICENSE](#), [UFE_ERR_LICENSE_NOT_MATCH](#),
[UFE_ERR_LICENSE_EXPIRED](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

phExtractor	[out] Pointer to handle of the extractor object
-------------	---

See Also

[UFE_Delete](#)

Examples

Visual C++

```
UFE_STATUS ufe_res;
HUFExtractor hExtractor;

ufe_res = UFE_Create(&hExtractor);
if(ufe_res == UFE_OK) {
    // UFE_Create is succeeded
} else {
    // UFE_Create is failed
    // Use UFE\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS
Dim hExtractor As Long

ufe_res = UFE_Create(hExtractor)
If (ufe_res = UFE_STATUS.OK) Then
```

```
' UFE_Create is succeeded
Else
    ' UFE_Create is failed
        ' Use UFE\_GetErrorString function to show error string
End If
```

Visual C++

```
int ufm_res;
Pointer hExtractor=null;
PointerByReference refExtractor = new PointerByReference();
UFExtractorClass libExtractor=null;

//load library(libExtractor)
libExtractor=
(UFExtractorClass )Native.loadLibrary("UFExtractor",UFExtractorClass.class);

ufm_res = libExtractor.UFE_Create(refExtractor);
if(ufm_res == libExtractor.UFE_OK) {
    // UFM_Create is succeeded
    hExtractor = refExtractor.getValue();
} else {
    // UFE_Create is failed
    // Use UFE\_GetErrorString function to show error string
}
```

UFE_Delete

Deletes specified extractor.

```
UFE_STATUS UFE_API UFE_Delete(
    HUFExtractor hExtractor
);

*java
int UFE_Delete(Pointer hExtractor);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
------------	-------------------------------------

See Also

[UFE_Create](#)

Examples

Visual C++

```
UFE_STATUS ufe_res;
HUFExtractor hExtractor;

// Create hExtractor and use

ufe_res = UFE_Delete(hExtractor);
if (ufe_res == UFE_OK) {
    // UFE_Delete is succeeded
} else {
    // UFE_Delete is failed
    // Use UFE\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS
Dim hExtractor As Long

' Create hExtractor and use
```

```
ufe_res = UFE_Delete(hExtractor)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_Delete is succeeded
Else
    ' UFE_Delete is failed
    ' Use UFE\_GetErrorString function to show error string
End If
```

```
java
int ufe_res;
Pointer hExtractor;

//load library(libExtractor)

// Create hExtractor and use

ufe_res = libExtractor.UFE_Delete(hExtractor);
if (ufe_res == libExtractor.UFE_OK) {
    // UFE_Delete is succeeded
} else {
    // UFE_Delete is failed
    // Use UFE\_GetErrorString function to show error string
}
```

UFE_GetMode

Gets mode of the specified extractor.

```
UFE_STATUS UFE_API UFE_GetMode(
    HUFExtractor hExtractor,
    int* pnMode
);

*java
int UFE_GetMode(
    Pointer hExtractor,
    IntByReference pnMode
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
pnMode	[out] Receives the mode of the specified extractor object

See also

[UFE_SetMode](#)

Examples

Visual C++
<pre>UFE_STATUS ufe_res; HUFExtractor hExtractor; int nMode; // Create hExtractor ufe_res = UFE_GetMode(hExtractor, &nMode); if(ufe_res == UFE_OK) { // UFE_GetMode is succeeded } else { // UFE_GetMode is failed // Use UFE_GetErrorString function to show error string }</pre>

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS
Dim hExtractor As Long
Dim Mode As Long

' Create hExtractor

ufe_res = UFE_GetMode(hExtractor, Mode)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_GetMode is succeeded
Else
    ' UFE_GetMode is failed
    ' Use UFE\_GetErrorString function to show error string
End If
```

java

```
int ufe_res;
Pointer hExtractor;
IntByReference refMode = new IntByReference();
int nMode;

//load library(libExtractor)

// Create hExtractor

ufe_res = libExtractor.UFE_GetMode(hExtractor, refMode);
if (ufe_res == libExtractor.UFE_OK) {
    // UFE_GetMode is succeeded
    nMode = refMode.getValue();
} else {
    // UFE_GetMode is failed
    // Use UFE\_GetErrorString function to show error string
}
```

UFE_SetMode

Sets mode of the specified extractor.

```
UFE_STATUS UFE_API UFE_SetMode(
    HUFExtractor hExtractor,
    int nMode
);

*java
int UFE_SetMode(
    Pointer hExtractor,
    int nMode
);
```

Possible return values

UFE_OK, UFE_ERROR, UFE_ERR_INVALID_PARAMETERS,
UFE_ERR_UNDEFINED_MODE

Parameters

<u>hExtractor</u>	[in] Handle of the extractor object
<u>nMode</u>	[in] Specifies the <u>mode</u>

See also

[UFE_GetMode](#)

Examples

Visual C++
<pre>UFE_STATUS ufe_res; HUFExtractor hExtractor; int nMode; // Create hExtractor // Set the mode for extractor, nMode ufe_res = UFE_SetMode(hExtractor, nMode); if(ufe_res == UFE_OK) { // UFE_SetMode is succeeded } else {</pre>

```
// UFE_SetMode is failed  
// Use UFE\_GetErrorString function to show error string  
}
```

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS  
Dim hExtractor As Long  
Dim Mode As Long  
  
' Create hExtractor  
  
' Set the mode for extractor, Mode  
  
ufe_res = UFE_SetMode(hExtractor, Mode)  
If (ufe_res = UFE_STATUS.OK) Then  
    ' UFE_SetMode is succeeded  
Else  
    ' UFE_SetMode is failed  
    ' Use UFE\_GetErrorString function to show error string  
End If
```

java

```
int ufe_res;  
Pointer hExtractor;  
int nMode;  
  
//load library(libExtractor)  
  
// Create hExtractor  
  
// Set the mode for extractor, nMode  
  
ufe_res = libExtractor.UFE_SetMode(hExtractor, nMode);  
if (ufe_res == libExtractor.UFE_OK) {  
    // UFE_SetMode is succeeded  
} else {  
    // UFE_SetMode is failed  
    // Use UFE\_GetErrorString function to show error string  
}
```

UFE_GetParameter

Gets parameter value.

```
UFE_STATUS UFE_API UFE_GetParameter(
    HUFExtractor hExtractor,
    int nParam,
    void* pValue
);

*java
int UFE_GetParameter(
    Pointer hExtractor,
    int nParam,
    PointerByReference pValue
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
nParam	[in] Parameter type; one of parameters
pValue	[out] Receives parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

See also

[UFE_SetParameter](#)

Examples

Visual C++

```
UFE_STATUS ufe_res;
HUFExtractor hExtractor;
int nValue;

// Create hExtractor

// Get detect core
ufe_res = UFE_GetParameter(hExtractor, UFE_PARAM_DETECT_CORE, &nValue);
```

```
// Error handling routine is omitted

// Get template size
ufe_res = UFE_GetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE, &nValue);
// Error handling routine is omitted

// Get use SIF
ufe_res = UFE_GetParameter(hExtractor, UFE_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted
```

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS
Dim hExtractor As Long
Dim nValue As Long

' Create hExtractor

' Get detect core
ufe_res = UFE_GetParameter(hExtractor, UFE_PARAM_DETECT_CORE, nValue)
' Error handling routine is omitted

' Get template size
ufe_res = UFE_GetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE, nValue)
' Error handling routine is omitted

' Get use SIF
ufe_res = UFE_GetParameter(hExtractor, UFE_PARAM_USE_SIF, nValue)
' Error handling routine is omitted
```

```
java
int ufe_res;
Pointer hExtractor;
PointerByReference refValue = new PointerByReference();

//load library(libExtractor)

// Create hExtractor

// Get detect core
ufe_res = libExtractor.UFE_GetParameter(hExtractor, UFE_PARAM_DETECT_CORE,
refValue);
// Error handling routine is omitted

// Get template size
ufe_res = libExtractorUFE_GetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE,
refValue);
// Error handling routine is omitted
```

```
// Get use SIF  
ufe_res = libExtractorUFE_GetParameter(hExtractor, UFE_PARAM_USE_SIF,  
refValue);  
// Error handling routine is omitted
```

UFE_SetParameter

Sets parameter value.

```
UFE_STATUS UFE_API UFE_SetParameter(
    HUFExtractor hExtractor,
    int nParam,
    void* pValue
);

*java
int UFE_SetParameter(
    Pointer hExtractor,
    int nParam,
    PointerByReference pValue
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
nParam	[in] Parameter type; one of parameters
pValue	[in] Pointer to parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

See also

[UFE_GetParameter](#)

Examples

Visual C++
<pre>UFE_STATUS ufe_res; HUFExtractor hExtractor; int nValue; // Create hExtractor // Set detect core to nValue ufe_res = UFE_SetParameter(hExtractor, UFE_PARAM_DETECT_CORE, &nValue);</pre>

```
// Error handling routine is omitted

// Set template size to nValue
ufe_res = UFE_SetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE, &nValue);
// Error handling routine is omitted

// Set use SIF to nValue
ufe_res = UFE_SetParameter(hExtractor, UFE_PARAM_USE_SIF, &nValue);
// Error handling routine is omitted
```

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS
Dim hExtractor As Long
Dim nValue As Long

' Create hExtractor

' Set detect core to nValue
ufe_res = UFE_SetParameter(hExtractor, UFE_PARAM_DETECT_CORE, nValue)
' Error handling routine is omitted

' Set template size to nValue
ufe_res = UFE_SetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE, nValue)
' Error handling routine is omitted

' Set use SIF to nValue
ufe_res = UFE_SetParameter(hExtractor, UFE_PARAM_USE_SIF, nValue)
' Error handling routine is omitted
```

java

```
int ufe_res;
Pointer hExtractor;
int nValue;

//load library(libExtractor)

// Create hExtractor

// Set detect core to nValue
ufe_res = libExtractor.UFE_SetParameter(hExtractor,
UFE_PARAM_DETECT_CORE,&nValue);
// Error handling routine is omitted

// Set template size to nValue
ufe_res = libExtractor.UFE_SetParameter(hExtractor, UFE_PARAM_TEMPLATE_SIZE,
nValue);
// Error handling routine is omitted
```

```
// Set use SIF to nValue  
ufe_res = libExtractor.UFE_SetParameter(hExtractor, UFE_PARAM_USE_SIF, nValue);  
// Error handling routine is omitted
```

UFE_DrawImage

Draws the fingerprint image which is acquired using [UFE_LoadImageFromBMPFile\(\)](#) or [UFE_LoadImageFromWSQFile\(\)](#)

This function is not supported on java.

```
UFS_STATUS UFS_API UFS_DrawCaptureImageBuffer(
    unsigned char* pImage,
    int nWidth,
    int nHeight,
    HDC hDC,
    int nLeft,
    int nTop,
    int nRight,
    int nBottom
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

pImage	[in] Pointer to raw image data
nWidth	[in] the width of the raw image data pImage
nHeight	[in] the height of the raw image data pImage
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

See also

[UFE_LoadImageFromBMPFile\(\)](#) , [UFE_LoadImageFromWSQFile\(\)](#)

Examples

Visual C++
UFE_STATUS ufe_res; unsigned char* Image;

```

int Width;
int Height;

// Load image from szFilename
ufe_res = UFE_LoadImageFromBMPFile(szFilename, Image, &Width, &Height);
if(ufe_res == UFE_OK) {
    // UFE_LoadImageFromBMPFile is succeeded
} else {
    // UFE_LoadImageFromBMPFile is failed
    // Use UFE\_GetErrorString function to show error string
}

// Get HDC and CRect determine rectangle to draw image, hDC, rect

UFE_DrawImage(Image, nWidth, nHeight, pDC->m_hDC, rect.left, rect.top, rect.right,
rect.bottom);
if(ufe_res == UFE_OK) {
    // UFE_DrawImage is succeeded
} else {
    // UFE_DrawImage is failed
    // Use UFE\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

Dim ufe_res As UFE_STATUS
Dim Image(MAX_IMAGE_WIDTH * MAX_IMAGE_HEIGHT) As Byte
Dim Width As Long
Dim Height As Long

' Load image from szFilename
ufe_res = UFE_LoadImageFromBMPFile(szFilename, Image, Width, Height)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_LoadImageFromBMPFile is succeeded
Else
    ' UFE_LoadImageFromBMPFile is failed
    ' Use UFE\_GetErrorString function to show error string
End If

' Use picture box Device Context which will draw loaded image
' ex) DC name is 'Frame'
ufe_res = UFE_DrawImage(Image(0), Width, Height, Frame.hDC, Frame.ScaleLeft,
Frame.ScaleTop, (Frame.ScaleLeft + Frame.ScaleWidth), (Frame.ScaleTop +
Frame.ScaleHeight))
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_DrawImage is succeeded
Else
    ' UFE_DrawImage is failed
    ' Use UFE\_GetErrorString function to show error string
End If

```

UFE_Extract

Extracts a template from the specified image.

```
UFE_STATUS UFE_API UFE_Extract(
    HUFExtractor hExtractor,
    unsigned char* pImage,
    int nWidth,
    int nHeight,
    int nResolution,
    unsigned char* pTemplate,
    int* pnTemplateSize,
    int* pnEnrollQuality
);

*java
int UFE_Extract(
    Pointer hExtractor,
    byte[] pImage,
    int nWidth,
    int nHeight,
    int nResolution,
    byte[] pTemplate,
    IntByReference pnTemplateSize,
    IntByReference pnEnrollQuality
);
```

Possible return values

UFE_OK, UFE_ERROR, UFE_ERR_LICENSE_NOT_MATCH,
UFE_ERR_LICENSE_EXPIRED, UFE_ERR_INVALID_PARAMETERS,
UFE_ERR_NOT_GOOD_IMAGE, UFE_ERR_EXTRACTION_FAILED,
UFE_ERR_CORE_NOT_DETECTED, UFE_ERR_CORE_TO_LEFT,
UFE_ERR_CORE_TO_LEFT_TOP, UFE_ERR_CORE_TO_TOP,
UFE_ERR_CORE_TO_RIGHT_TOP, UFE_ERR_CORE_TO_RIGHT,
UFE_ERR_CORE_TO_RIGHT_BOTTOM, UFE_ERR_CORE_TO_BOTTOM,
UFE_ERR_CORE_TO_LEFT_BOTTOM

Parameters

hExtractor	[in] Handle of the extractor object
pImage	[in] Point to buffer storing input image; input image is assumed to be top-down order and 8 bit gray for pixel format
nWidth	[in] Width of the input image; nWidth must be less than 640
nHeight	[in] Height of the input image; nHeight must be less than 640
nResolution	[in] Resolution of the input image

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.

Examples

Visual C++

```
// Assume template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384

UFE_STATUS ufe_res;
HUFExtractor hExtractor;
unsigned char* pImage;
int nWidth;
int nHeight;
int nResolution;
unsigned char Template[MAX_TEMPLATE_SIZE];
int nTemplateSize;
int nEnrollQuality;

// Create hExtractor

// Get input image to pImage, nWidth, nHeight, nResolution

ufe_res = UFE_Extract(hExtractor, pImage, nWidth, nHeight, nResolution, Template,
nTemplateSize, &nEnrollQuality);
if(ufe_res == UFE_OK) {
    // UFE_Extract is succeeded
    // Check nEnrollQuality
} else {
    // UFE_Extract is failed
    // Use UFE\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufe_res As UFE_STATUS
Dim hExtractor As Long
Dim Image() As Byte
Dim Width As Long
Dim Height As Long
Dim Resolution As Long
```

```

Dim Template(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateSize As Long
Dim EnrollQuality As Long

' Create hExtractor

' Get input image to Image, Width, Height, Resolution

ufe_res = UFE_Extract(hExtractor, Image(0), Width, Height, Resolution, Template(0),
TemplateSize, EnrollQuality)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_Extract is succeeded
    ' Check EnrollQuality
Else
    ' UFE_Extract is failed
    ' Use UFE\_GetErrorString function to show error string
End If

```

```

java
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufe_res;
Pointer hExtractor;
byte pImage = new byte[1024*10];
int nWidth;
int nHeight;
int nResolution;
int nQC;
byte[] Template = new byte[MAX_TEMPLATE_SIZE];
IntByReference nTemplateSize = new IntByReference();
IntByReference nEnrollQuality = new IntByReference();

//load library(libExtractor)

// Create hExtractor

// Get input image to pImage, nWidth, nHeight, nResolution

ufe_res = libExtractor.UFE_Extract(hExtractor, pImage, nWidth, nHeight, nResolution,
Template, nTemplateSize, nEnrollQuality);
if (ufe_res == libExtractor.UFE_OK) {
    // UFE_Extract is succeeded
    // Check nEnrollQuality
    nQC=nEnrollQuality.getValue();
} else {
    // UFE_Extract is failed
    // Use UFE\_GetErrorString function to show error string
}

```

UFE_SetEncryptionKey

Sets encryption key.

```
UFE_STATUS UFE_API UFE_SetEncryptionKey(
    HUFExtractor hExtractor,
    unsigned char* pKey
);

/*java
int UFE_SetEncryptionKey(
    Pointer hExtractor,
    String pKey
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
pKey	[in] Pointer to the 32 bytes key array

See also

[UFE_EncryptTemplate](#), [UFE_DecryptTemplate](#)

Examples

Visual C++
<pre>UFE_STATUS ufe_res; HUFExtractor hExtractor; unsigned char UserKey[32]; // Get hScanner handle // Generate 32 byte encryption key to UserKey ufe_res = UFE_SetEncryptionKey(hExtractor, UserKey); if(ufe_res == UFE_OK) { // UFE_SetEncryptionKey is succeeded } else { // UFE_SetEncryptionKey is failed // Use UFE_GetErrorString function to show error string }</pre>

```
}
```

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS
Dim hExtractor As Long
Dim UserKey(32 - 1) As Byte

' Get hScanner handle

' Generate 32 byte encryption key to UserKey

ufe_res = UFE_SetEncryptionKey(hExtractor, UserKey)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_SetEncryptionKey is succeeded
Else
    ' UFE_SetEncryptionKey is failed
    ' Use UFE_GetErrorString function to show error string
End If
```

java

```
int ufe_res;
Pointer hExtractor;
String UserKey;

//load library(libExtractor)

// Get hScanner handle

// Generate 32 byte encryption key to UserKey

ufe_res = libExtractor.UFE_SetEncryptionKey(hExtractor, UserKey);
if (ufe_res == libExtractor.UFE_OK) {
    // UFE_SetEncryptionKey is succeeded
} else {
    // UFE_SetEncryptionKey is failed
    // Use UFE_GetErrorString function to show error string
}
```

UFE_EncryptTemplate

Encrypts template.

```
UFE\_STATUS UFE_API UFE_EncryptTemplate(
    HUFExtractor hExtractor,
    unsigned char* pTemplateInput,
    int nTemplateSize,
    unsigned char* pTemplateOutput,
    int\* pnTemplateOutputSize
);

*java
int UFE_EncryptTemplate(
    Pointer hExtractor,
    byte[] pTemplateInput,
    int nTemplateSize,
    byte[] pTemplateOutput,
    IntByReference pnTemplateOutputSize
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
pTemplateInput	[in] Pointer to input template array
nTemplateInputSize	[in] Input template size
pTemplateOutput	[out] Pointer to output template array
pnTemplateOutputSize	[out] Receives output template size

See also

[UFE_SetEncryptionKey](#), [UFE_DecryptTemplate](#)

Examples

Visual C++
// Assume template size is 384 bytes <code>#define MAX_TEMPLATE_SIZE 384</code>

```

UFE_STATUS ufe_res;
HUFExtractor hExtractor;
unsigned char TemplateInput[MAX_TEMPLATE_SIZE];
unsigned char TemplateOutput[MAX_TEMPLATE_SIZE];
int TemplateInputSize;
int TemplateOutputSize;

// Get hExtractor handle

// Get an input template to encrypt, TemplateInput and TemplateInputSize

// Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE;

ufe_res = UFE_EncryptTemplate(hExtractor, TemplateInput, TemplateInputSize,
TemplateOutput, &TemplateOutputSize);
if (ufe_res == UFE_OK) {
    // UFE_EncryptTemplate is succeeded
} else {
    // UFE_EncryptTemplate is failed
    // Use UFE\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufe_res As UFE_STATUS
Dim hExtractor As Long
Dim TemplateInput(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateOutput(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateInputSize As Long
Dim TemplateOutputSize As Long

' Get hExtractor handle

' Get an input template to encrypt, TemplateInput and TemplateInputSize

' Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE

ufe_res = UFE_EncryptTemplate(hExtractor, TemplateInput(0), TemplateInputSize,
TemplateOutput(0), TemplateOutputSize)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_EncryptTemplate is succeeded
Else
    ' UFE_EncryptTemplate is failed
    ' Use UFE\_GetErrorString function to show error string
End If

```

```
java
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufs_res;
Pointer hExtractor;
byte[] TemplateInput = new byte[MAX_TEMPLATE_SIZE];
byte[] TemplateOutput = new byte[MAX_TEMPLATE_SIZE];
int TemplateInputSize;
IntByReference TemplateOutputSize = new IntByReference();;

//load library(libExtractor)

// Get hExtractor handle

// Get an input template to encrypt, TemplateInput and TemplateInputSize

// Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE;

ufs_res = libExtractor.UFE_EncryptTemplate(hExtractor, TemplateInput,
TemplateInputSize, TemplateOutput, TemplateOutputSize);
if(ufs_res == libExtractor.UFE_OK) {
    // UFE_EncryptTemplate is succeeded
} else {
    // UFE_EncryptTemplate is failed
    // Use UFE\_GetErrorString function to show error string
}
```

UFE_DecryptTemplate

Decrypts template.

```
UFE_STATUS UFE_API UFE_DecryptTemplate(
    HUFExtractor hExtractor,
    unsigned char* pTemplateInput,
    int nTemplateSize,
    unsigned char* pTemplateOutput,
    int* pnTemplateOutputSize
);

int UFE_DecryptTemplate(
    Pointer hExtractor,
    byte[] pTemplateInput,
    int nTemplateSize,
    byte[] pTemplateOutput,
    IntByReference pnTemplateOutputSize
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
pTemplateInput	[in] Pointer to input template array
nTemplateInputSize	[in] Input template size
pTemplateOutput	[out] Pointer to output template array
pnTemplateOutputSize	[out] Receives output template size

See also

[UFE_SetEncryptionKey](#), [UFE_EncryptTemplate](#)

Examples

Visual C++

```
// Assume template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384

UFE_STATUS ufe_res;
```

```

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

// Get hExtractor handle

// Get an encrypted template, TemplateInput and TemplateInputSize

// Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE;

ufe_res = UFE_DecryptTemplate(hExtractor, TemplateInput, TemplateInputSize,
TemplateOutput, &TemplateOutputSize);
if(ufe_res == UFE_OK) {
    // UFE_DecryptTemplate is succeeded
} else {
    // UFE_DecryptTemplate is failed
    // Use UFE\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

' Assume template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384

Dim ufe_res As UFE_STATUS
Dim hExtractor As Long
Dim TemplateInput(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateOutput(MAX_TEMPLATE_SIZE - 1) As Byte
Dim TemplateInputSize As Long
Dim TemplateOutputSize As Long

' Get hExtractor handle

' Get an encrypted template, TemplateInput and TemplateInputSize

' Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE

ufe_res = UFE_DecryptTemplate(hExtractor, TemplateInput(0), TemplateInputSize,
TemplateOutput(0), TemplateOutputSize)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_DecryptTemplate is succeeded
Else
    ' UFE_DecryptTemplate is failed
    ' Use UFE\_GetErrorString function to show error string
End If

```

```
java
// Assume template size is 384 bytes
int MAX_TEMPLATE_SIZE = 384;

int ufe_res;
Pointer hExtractor;
byte[] TemplateInput = new byte[MAX_TEMPLATE_SIZE];
byte[] TemplateOutput = new byte[MAX_TEMPLATE_SIZE];
int TemplateInputSize;
IntByReference TemplateOutputSize = new IntByReference();

//load libray(libExtractor)

// Get hExtractor handle

// Get an encrypted template, TemplateInput and TemplateInputSize

// Set output template buffer size
TemplateoutputSize = MAX_TEMPLATE_SIZE;

ufs_res = libExtractor.UFE_DecryptTemplate(hExtractor, TemplateInput,
TemplateInputSize, TemplateOutput, TemplateOutputSize);
if(ufe_res == libExtractor.UFE_OK) {
    // UFE_DecryptTemplate is succeeded
} else {
    // UFE_DecryptTemplate is failed
    // Use UFE\_GetErrorString functionto show error string
}
```

UFE_LoadImageFromBMPFile

Loads raw image data from a bitmap file.

```
UFE\_STATUS UFE_API UFE_LoadImageFromBMPFile(
    const char* szBMPFileName,
    unsigned char* pImage,
    int* pnWidth,
    int* pnHeight
);

*java
int UFE_LoadImageFromBMPFile(
    String szBMPFileName,
    byte[] pImage,
    IntByReference pnWidth,
    IntByReference pnHeight
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

szBMPFileName	[in] Specifies input bitmap file name
pImage	[out] Pointer to raw image data
pnWidth	[out] Receives the width of the raw image data pImage
pnHeight	[out] Receives the height of the raw image data pImage

Examples

Visual C++

```
// Assume maximum image size is 1024 x 1024
#define MAX_IMAGE_WIDTH 1024
#define MAX_IMAGE_HEIGHT 1024

UFE_STATUS ufe_res;
char szFileName[256];
unsigned char* pImage;
int nWidth;
int nHeight;
```

```

// Get file name from user, szFileName

// Allocate image buffer
pImage = (unsigned char*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT);

ufe_res = UFE_LoadImageFromBMPFile(szFileName, pImage, &nWidth, &nHeight);
if (ufe_res == UFE_OK) {
    // UFE_LoadImageFromBMPFile is succeeded
} else {
    // UFE_LoadImageFromBMPFile is failed
    // Use UFE\_GetErrorString function to show error string
}

// Use image buffer

// Free image buffer
free(pImage)

```

Visual Basic 6.0

```

' Assume maximum image size is 1024 x 1024
Const MAX_IMAGE_WIDTH As Long = 1024
Const MAX_IMAGE_HEIGHT As Long = 1024

Dim ufe_res As UFE_STATUS
Dim szFileName As String
Dim Image(MAX_IMAGE_WIDTH * MAX_IMAGE_HEIGHT) As Byte
Dim Width As Long
Dim Height As Long

' Get file name from user, FileName

ufe_res = UFE_LoadImageFromBMPFile(FileName, Image(0), Width, Height)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_LoadImageFromBMPFile is succeeded
Else
    ' UFE_LoadImageFromBMPFile is failed
    ' Use UFE\_GetErrorString function to show error string
End If

```

java

```

// Assume maximum image size is 1024 x 1024
int MAX_IMAGE_WIDTH = 1024;
int MAX_IMAGE_HEIGHT = 1024;

int ufe_res;
String szFileName;
byte[] pImage = new byte[1024*10];
int nWidth;

```

```
int nHeight;

// Get file name from user, szFileName

// Allocate image buffer
pImage = (unsigned char*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT);

ufe_res = UFE_LoadImageFromBMPFile(szFileName, pImage, &nWidth, &nHeight);
if(ufe_res == UFE_OK) {
    // UFE_LoadImageFromBMPFile is succeeded
} else {
    // UFE_LoadImageFromBMPFile is failed
    // Use UFE\_GetErrorString function to show error string
}

// Use image buffer

// Free image buffer
free(pImage)
```

UFE_LoadImageFromBMPBuffer

Loads raw image data from a bitmap buffer.

```
UFE_STATUS UFE_API UFE_LoadImageFromBMPBuffer(
    unsigned char* pBMPBuffer,
    int nBMPBufferSize
    unsigned char* pImage,
    int* pnWidth,
    int* pnHeight
);

*java
int UFE_LoadImageFromBMPBuffer(
    byte[] pBMPBuffer,
    int nBMPBufferSize,
    byte[] pImage,
    IntByReference pnWidth,
    IntByReference pnHeight
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

pBMPBuffer	[in] Pointer to the input bitmap buffer
nBMPBufferSize	[in] Specifies the size of the input bitmap buffer
pImage	[out] Pointer to raw image data
pnWidth	[out] Receives the width of the raw image data pImage
pnHeight	[out] Receives the height of the raw image data pImage

Examples

Visual C++

```
// Assume maximum image size is 1024 x 1024
#define MAX_IMAGE_WIDTH 1024
#define MAX_IMAGE_HEIGHT 1024

UFE_STATUS ufe_res;
unsigned char* pBMPBuffer;
int nBMPBufferSize;
```

```
unsigned char* pImage;
int nWidth;
int nHeight;

// Get bitmap buffer, pBMPBuffer, nBMPBufferSize

// Allocate image buffer
pImage = (unsigned char*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT);

ufe_res = UFE_LoadImageFromBMPBuffer(pBMPBuffer, nBMPBufferSize, pImage,
&nWidth, &nHeight);
if (ufe_res == UFE_OK) {
    // UFE_LoadImageFromBMPBuffer is succeeded
} else {
    // UFE_LoadImageFromBMPBuffer is failed
    // Use UFE\_GetErrorString function to show error string
}

// Use image buffer

// Free image buffer
free(pImage)
```

Visual Basic 6.0

```
' Assume maximum image size is 1024 x 1024
Const MAX_IMAGE_WIDTH As Long = 1024
Const MAX_IMAGE_HEIGHT As Long = 1024

Dim ufe_res As UFE_STATUS
Dim szFileName As String
Dim BMPBuffer() As Byte
Dim BMPBufferSize As Long
Dim Image(MAX_IMAGE_WIDTH * MAX_IMAGE_HEIGHT) As Byte
Dim Width As Long
Dim Height As Long

' Get bitmap buffer, BMPBuffer, BMPBufferSize

ufe_res = UFE_LoadImageFromBMPBuffer(BMPBuffer(0), BMPBufferSize, Image(0),
Width, Height)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_LoadImageFromBMPBuffer is succeeded
Else
    ' UFE_LoadImageFromBMPBuffer is failed
    ' Use UFE\_GetErrorString function to show error string
End If
```

java

```
// Assume maximum image size is 1024 x 1024
int MAX_IMAGE_WIDTH = 1024;
int MAX_IMAGE_HEIGHT = 1024;

int ufe_res;
byte[] pBMPBuffer = new byte[1024*10];
int nBMPBufferSize;
IntByReference nWidth = new IntByReference();
IntByReference nHeight = new IntByReference();

//load library(libExtractor)

// Get bitmap buffer, pBMPBuffer, nBMPBufferSize

// Allocate image buffer
byte[] pImage = new byte[MAX_IMAGE_WIDTH*MAX_IMAGE_HEIGHT];

ufe_res = libExtractor.UFE_LoadImageFromBMPBuffer(pBMPBuffer, nBMPBufferSize,
pImage, nWidth, nHeight);
if(ufe_res == libExtractor.UFE_OK) {
    // UFE_LoadImageFromBMPBuffer is succeeded
} else {
    // UFE_LoadImageFromBMPBuffer is failed
    // Use UFE_GetErrorString function to show error string
}

// Use image buffer
```

UFE_LoadImageFromWSQFile

Loads raw image data from a wsq file.
This function is not supported in JAVA yet.

```
UFE_STATUS UFE_API UFE_UFE_LoadImageFromWSQFile(
    const char* szWSQFileName,
    unsigned char* pImage,
    int* pnWidth,
    int* pnHeight
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

szWSQFileName	[in] Specifies input wsq file name
pImage	[out] Pointer to raw image data
pnWidth	[out] Receives the width of the raw image data pImage
pnHeight	[out] Receives the height of the raw image data pImage

Examples

Visual C++
<pre>// Assume maximum image size is 1024 x 1024 #define MAX_IMAGE_WIDTH 1024 #define MAX_IMAGE_HEIGHT 1024 UFE_STATUS ufe_res; char szFileName[256]; unsigned char* pImage; int nWidth; int nHeight; // Get file name from user, szFileName // Allocate image buffer pImage = (unsigned char*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT); ufe_res = UFE_LoadImageFromWSQFile(szFileName, pImage, &nWidth, &nHeight); if (ufe_res == UFE_OK) { // UFE_LoadImageFromWSQFile is succeeded } else {</pre>

```

    // UFE_LoadImageFromWSQFile is failed
    // Use UFE\_GetErrorString function to show error string
}

// Use image buffer

// Free image buffer
free(pImage)

```

Visual Basic 6.0

```

' Assume maximum image size is 1024 x 1024
Const MAX_IMAGE_WIDTH As Long = 1024
Const MAX_IMAGE_HEIGHT As Long = 1024

Dim ufe_res As UFE_STATUS
Dim szFileName As String
Dim Image(MAX_IMAGE_WIDTH * MAX_IMAGE_HEIGHT) As Byte
Dim Width As Long
Dim Height As Long

' Get file name from user, FileName

ufe_res = UFE_LoadImageFromWSQFile(FileName, Image(0), Width, Height)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_LoadImageFromWSQFile is succeeded
Else
    ' UFE_LoadImageFromWSQFile is failed
    ' Use UFE\_GetErrorString function to show error string
End If

```

UFE_LoadImageFromWSQBuffer

Loads raw image data from a wsq buffer.

```
UFE\_STATUS UFE_API UFE_LoadImageFromWSQBuffer(
    unsigned char\* pWSQBuffer,
    int nWSQBufferSize
    unsigned char\* pImage,
    int\* pnWidth,
    int\* pnHeight
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

pWSQBuffer	[in] Pointer to the input wsq buffer
nWSQBufferSize	[in] Specifies the size of the input wsq buffer
pImage	[out] Pointer to raw image data
pnWidth	[out] Receives the width of the raw image data pImage
pnHeight	[out] Receives the height of the raw image data pImage

Examples

Visual C++

```
// Assume maximum image size is 1024 x 1024
#define MAX_IMAGE_WIDTH 1024
#define MAX_IMAGE_HEIGHT 1024

UFE_STATUS ufe_res;
unsigned char\* pBMPBuffer;
int nBMPBufferSize;
unsigned char\* pImage;
int nWidth;
int nHeight;

// Get bitmap buffer, pWSQBuffer, nWSQBufferSize

// Allocate image buffer
pImage = (unsigned char\*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT);
```

```

ufe_res = UFE_LoadImageFromWSQBuffer(pWSQBuffer, nWSQBufferSize, pImage,
&nWidth, &nHeight);
if(ufe_res == UFE_OK) {
    // UFE_LoadImageFromWSQBuffer is succeeded
} else {
    // UFE_LoadImageFromWSQBuffer is failed
    // Use UFE\_GetErrorString function to show error string
}

// Use image buffer

// Free image buffer
free(pImage)

```

Visual Basic 6.0

```

' Assume maximum image size is 1024 x 1024
Const MAX_IMAGE_WIDTH As Long = 1024
Const MAX_IMAGE_HEIGHT As Long = 1024

Dim ufe_res As UFE_STATUS
Dim szFileName As String
Dim WSQBuffer() As Byte
Dim WSQBufferSize As Long
Dim Image(MAX_IMAGE_WIDTH * MAX_IMAGE_HEIGHT) As Byte
Dim Width As Long
Dim Height As Long

' Get bitmap buffer, WSQBuffer, WSQBufferSize

ufe_res = UFE_LoadImageFromWSQBuffer(WSQBuffer(0), WSQBufferSize, Image(0),
Width, Height)
If (ufe_res = UFE_STATUS.OK) Then
    ' UFE_LoadImageFromWSQBuffer is succeeded
Else
    ' UFE_LoadImageFromWSQBuffer is failed
    ' Use UFE\_GetErrorString function to show error string
End If

```

UFE_GetErrorString

Gets the error string for specified [UFE_STAUS](#) value.

```
UFE\_STATUS UFE_API UFE_GetErrorString(
    UFE\_STATUS res,
    char* szErrorString
);

*java
int UFE_GetErrorString(int res, byte[] szErrorString);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

res	[in] Status return value
szErrorString	[out] Receives error sting

Examples

Visual C++

```
UFE_STATUS ufe_res;
char strError[128];

// Get status return code, ufe_res

ufe_res = UFE_GetErrorString(ufe_res, strError);
if(ufe_res == UFE_OK) {
    // UFE_GetErrorString is succeeded
} else {
    // UFE_GetErrorString is failed
}
```

Visual Basic 6.0

```
Dim ufe_res As UFE_STATUS
Dim m_strError As String

' Get status return code, ufe_res

ufe_res = UFE_GetErrorString(ufe_res, strError)
If (ufe_res = UFE_STATUS.OK) Then
```

```
' UFE_GetErrorString is succeeded
Else
    ' UFE_GetErrorString is failed
End If
```

```
java
int ufe_res;
byte[] strError = new byte[128];

// Get status return code, ufe_res

ufe_res = libExtractor.UFE_GetErrorString(ufe_res, strError);
if(ufe_res == libExtractor.UFE_OK) {
    // UFE_GetErrorString is succeeded
} else {
    // UFE_GetErrorString is failed
}
```

UFE_GetTemplateType

Gets parameter value.

```
UFE_STATUS UFE_API UFE_GetTemplateType(
    HUFExtractor hExtractor,
    void* pValue
);

/*java
int UFE_GetTemplateType(
    Pointer hExtractor,
    IntByReference pValue
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle to the extractor object
pValue	[out] Receives parameter value of specified parameter type; pValue must point to adequate storage type matched to parameter type

See also

[UFE_SetTemplateType](#)

Examples

Visual C++
<pre>UFE_STATUS ufe_res; HUFExtractor hExtractor; int nValue; // Get hExtractor handle ufe_res = UFE_GetTemplateType(hExtractor,&nValue); // Error handling routine is omitted</pre>

Visual Basic 6.0
<pre>Dim ufe_res As UFE_STATUS</pre>

```
Dim hExtractor As Long
Dim nValue As Long

' Get hExtractor handle

ufe_res = UFE_GetTemplateType(hExtractor,nValue)
' Error handling routine is omitted
```

```
java
int ufe_res;
Pointer hExtractor;
int nTemplateType;
IntByReference refValue = new IntByReference();

//load library(libExtractor)

// Get hExtractor handle

ufe_res = libExtractor.UFE_GetTemplateType(hExtractor, refValue );
nTemplateType=refValue.getValue();
// Error handling routine is omitted
```

UFE_GetImageBufferTo19794_4ImageBuffer

Make an ISO 19794-4 image from raw image data.

```
UFE\_STATUS UFE_API UFE_GetImageBufferTo19794_4ImageBuffer(
    HUFExtractor hExtractor,
    unsigned char* pImageDataIn,
    int Width,
    int Height,
    unsigned char* pImageDataOut,
    int* pImageDataLength
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

<code>hExtractor</code>	[in] Handle of the extractor object
<code>pImageDataIn</code>	[in] Pointer to raw image data
<code>Width</code>	[in] Width of the raw image data
<code>Height</code>	[in] Height of the raw image data
<code>pImageDataOut</code>	[out] Pointer to output image data; ISO 19794-4 format
<code>pImageDataLength</code>	[out] length of output image data

Examples

Visual C++

```
// Assume maximum image size is 1024 x 1024
#define MAX_IMAGE_WIDTH 1024
#define MAX_IMAGE_HEIGHT 1024

UFE_STATUS ufe_res;
unsigned char* pImage;
int nWidth;
int nHeight;
// Make a buffer for output image
unsigned char pImageDataOut[100000];
int pImageDataLength;

// Allocate an image buffer
pImage = (unsigned char*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT);
```

```
ufe_res = UFE_LoadImageFromBMPFile(szFileName, pImage, &nWidth, &nHeight);
if(ufe_res == UFE_OK) {
    // UFE_LoadImageFromBMPBuffer is succeeded
} else {
    // UFE_LoadImageFromBMPBuffer is failed
    // Use UFE\_GetErrorString function to show error string
}

HUFExtractor hExtractor;

// Create extractor
ufe_res = UFE_Create(&hExtractor);
if(ufe_res == UFE_OK) {
    // UFE_Create is succeeded
}
else
{
    // UFE_Create is failed
    return;
}

UFE_GetImageBufferTo19794_4ImageBuffer(hExtractor, pImage, nWidth, nHeight,
pImageDataOut, &pImageDataLength);

// If want to save the image as a file, use C function, which saves the binary data.
FILE* fp;
fp = fopen("File Name", "wb");
fwrite(pImageDataOut, 1, pImageDataLength, fp);
fclose(fp);

// Free image buffer
free(pImage)
```

UFE_GetImageBufferToBMPImageBuffer

Make a BMP image from raw image data.

```
UFE\_STATUS UFE_API UFE_GetImageBufferToBMPImageBuffer(
    HUFExtractor hExtractor,
    unsigned char\* pImageDataIn,
    int Width,
    int Height,
    unsigned char\* pImageDataOut,
    int\* pImageDataLength
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

<code>hExtractor</code>	[in] Handle of the extractor object
<code>pImageDataIn</code>	[in] Pointer to raw image data
<code>Width</code>	[in] Width of the raw image data
<code>Height</code>	[in] Height of the raw image data
<code>pImageDataOut</code>	[out] Pointer to output image data; BMP format
<code>pImageDataLength</code>	[out] length of output image data

Examples

Visual C++

```
// Assume maximum image size is 1024 x 1024
#define MAX_IMAGE_WIDTH 1024
#define MAX_IMAGE_HEIGHT 1024

UFE_STATUS ufe_res;
unsigned char\* pImage;
int nWidth;
int nHeight;
// Make a buffer for output image
unsigned char pImageDataOut[100000];
int pImageDataLength;

// Allocate an image buffer
pImage = (unsigned char\*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT);
```

```
ufe_res = UFE_LoadImageFromBMPFile(szFileName, pImage, &nWidth, &nHeight);
if(ufe_res == UFE_OK) {
    // UFE_LoadImageFromBMPBuffer is succeeded
} else {
    // UFE_LoadImageFromBMPBuffer is failed
    // Use UFE\_GetErrorString function to show error string
}

HUFExtractor hExtractor;

// Create extractor
ufe_res = UFE_Create(&hExtractor);
if(ufe_res == UFE_OK) {
    // UFE_Create is succeeded
}
else
{
    // UFE_Create is failed
    return;
}

UFE_GetImageBufferToBMPIImageBuffer(hExtractor, pImage, nWidth, nHeight,
pImageDataOut, &pImageDataLength);

// If want to save the image as a file, use C function, which saves the binary data.
FILE* fp;
fp = fopen("File Name", "wb");
fwrite(pImageDataOut, 1, pImageDataLength, fp);
fclose(fp);

// Free image buffer
free(pImage)
```

UFE_GetImageBufferToJPEGImageBuffer

Make a JPEG image from raw image data.

```
UFE_STATUS UFE_API UFE_GetImageBufferToJPEGImageBuffer(
    HUFExtractor hExtractor,
    unsigned char* pImageDataIn,
    int Width,
    int Height,
    unsigned char* pImageDataOut,
    int* pImageDataLength
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
pImageDataIn	[in] Pointer to raw image data
Width	[in] Width of the raw image data
Height	[in] Height of the raw image data
pImageDataOut	[out] Pointer to output image data; JPEG format
pImageDataLength	[out] length of output image data

Examples

Visual C++

```
// Assume maximum image size is 1024 x 1024
#define MAX_IMAGE_WIDTH 1024
#define MAX_IMAGE_HEIGHT 1024

UFE_STATUS ufe_res;
unsigned char* pImage;
int nWidth;
int nHeight;
// Make a buffer for output image
unsigned char pImageDataOut[100000];
int pImageDataLength;

// Allocate an image buffer
pImage = (unsigned char*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT);
```

```
ufe_res = UFE_LoadImageFromBMPFile(szFileName, pImage, &nWidth, &nHeight);
if(ufe_res == UFE_OK) {
    // UFE_LoadImageFromBMPBuffer is succeeded
} else {
    // UFE_LoadImageFromBMPBuffer is failed
    // Use UFE\_GetErrorString function to show error string
}

HUFExtractor hExtractor;

// Create extractor
ufe_res = UFE_Create(&hExtractor);
if(ufe_res == UFE_OK) {
    // UFE_Create is succeeded
}
else
{
    // UFE_Create is failed
    return;
}

UFE_GetImageBufferToJPEGImageBuffer(hExtractor, pImage, nWidth, nHeight,
pImageDataOut, &pImageDataLength);

// If want to save the image as a file, use C function, which saves the binary data.
FILE* fp;
fp = fopen("File Name", "wb");
fwrite(pImageDataOut, 1, pImageDataLength, fp);
fclose(fp);

// Free image buffer
free(pImage)
```

UFE_GetImageBufferToJP2ImageBuffer

Make a JP2 image from raw image data.

```
UFE\_STATUS UFE_API UFE_GetImageBufferToJP2ImageBuffer(
    HUFExtractor hExtractor,
    unsigned char* pImageDataIn,
    int Width,
    int Height,
    unsigned char* pImageDataOut,
    int* pImageDataLength
);
```

Possible return values

[UFE_OK](#), [UFE_ERROR](#), [UFE_ERR_INVALID_PARAMETERS](#)

Parameters

hExtractor	[in] Handle of the extractor object
pImageDataIn	[in] Pointer to raw image data
Width	[in] Width of the raw image data
Height	[in] Height of the raw image data
pImageDataOut	[out] Pointer to output image data; JPEG2000 format
pImageDataLength	[out] length of output image data

Examples

Visual C++

```
// Assume maximum image size is 1024 x 1024
#define MAX_IMAGE_WIDTH 1024
#define MAX_IMAGE_HEIGHT 1024

UFE_STATUS ufe_res;
unsigned char* pImage;
int nWidth;
int nHeight;
// Make a buffer for output image
unsigned char pImageDataOut[100000];
int pImageDataLength;

// Allocate an image buffer
pImage = (unsigned char*)malloc(MAX_IMAGE_WIDTH, MAX_IMAGE_HEIGHT);
```

```
ufe_res = UFE_LoadImageFromBMPFile(szFileName, pImage, &nWidth, &nHeight);
if(ufe_res == UFE_OK) {
    // UFE_LoadImageFromBMPBuffer is succeeded
} else {
    // UFE_LoadImageFromBMPBuffer is failed
    // Use UFE\_GetErrorString function to show error string
}

HUFExtractor hExtractor;

// Create extractor
ufe_res = UFE_Create(&hExtractor);
if(ufe_res == UFE_OK) {
    // UFE_Create is succeeded
}
else
{
    // UFE_Create is failed
    return;
}

UFE_GetImageBufferToJP2ImageBuffer(hExtractor, pImage, nWidth, nHeight,
pImageDataOut, &pImageDataLength);

// If want to save the image as a file, use C function, which saves the binary data.
FILE* fp;
fp = fopen("File Name","wb");
fwrite(pImageDataOut,1, pImageDataLength,fp);
fclose(fp);

// Free image buffer
free(pImage)
```

UFDatabase module

UFDatabase module provides functionality for managing database, adding / updating / removing / getting templates with user data, etc.

Requirements

Visual C++
<ul style="list-style-type: none"> Required header: include\UFDatabase.h Required lib: lib\UFDatabase.lib Required dll: bin\UFDatabase.dll
Visual Basic 6.0
<ul style="list-style-type: none"> Required reference: Suprema type library (bin\Suprema.tlb) Required dll: bin\UFDatabase.dll

Database table structure

UFDatabase module uses predefined table design named as 'Fingerprints'. Although a template database file (bin\UFDatabase.mdb) is provided, users can create database table for UFDatabase module using following information.

Table name: Fingerprints			
Field name	Data format	Field size	Remarks
Serial	Serial number	Long integer	Serial Number for Indexing DB (Primary key)
UserID	Text	50	User ID Information (Mandatory)
FingerIndex	Number	Long integer	Finger Index for Identifying finger (Mandatory)
Template1	OLE object	(1024)	Fingerprint Template Data (Mandatory)
Template2	OLE object	(1024)	Fingerprint Template Data (Optional)
Memo	Text	100	Additional Data (Optional)

Definitions

Status return value (UFD_STATUS)

Every function in UFDatabase module returns UFD_STATUS (integer) value having one of following values,

Status value definition	Code	Meaning
UFD_OK	0	Success
UFD_ERROR	-1	General error
UFD_ERR_NO_LICENSE	-101	System has no license
UFD_ERR_LICENSE_NOT_MATCH	-102	License is not match
UFD_ERR_LICENSE_EXPIRED	-103	License is expired
UFD_ERR_NOT_SUPPORTED	-111	This function is not supported
UFD_ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
UFD_ERR_SAME_FINGER_EXIST	-501	Same finger exists on database

Database handle

HUFDatabase defines handle to UFDatabase object.

```
typedef void* HUFDatabase;
```

UFD_Open

Opens a database using specified connection string.

```
UFD_STATUS UFD_API UFD_Open(
    const char* szConnection,
    const char* szUserID,
    const char* szPassword,
    HUFDatabase* phDatabase
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

szConnection	[in] Specifies ADO connection strings; to connect to an Access file using the JET OLE DB Provider, use "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=mdb_file_path;"; if database is password protected, use "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=mdb_file_path;Jet OLEDB:Database Password=mdb_password;"
szUserID	[in] Specifies user ID directly passed to ADO open method (can be NULL)
szPassword	[in] Specifies password directly passed to ADO open method (can be NULL)
phDatabase	[out] Pointer to handle of the database object

See also

[UFD_Close](#)

Examples

Visual C++

```
UFD_STATUS ufd_res;
HUFDatabase hDatabase;

ufd_res = UFD_Open("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=UFDDatabase.mdb;", NULL, NULL, &hDatabase);
if(ufd_res == UFD_OK) {
    // UFD_Open is succeeded
} else {
    // UFD_Open is failed
    // Use UFD\_GetErrorString function to show error string
```

{

Visual Basic 6.0

```
Dim ufd_res As UFD_STATUS
Dim hDatabase As Long

ufd_res = UFD_Open("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=UFDatabase.mdb;", "", "", hDatabase)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_Open is succeeded
Else
    ' UFD_Open is failed
    ' Use UFD\_GetErrorString function to show error string
End If
```

UFD_Close

Closes specified database.

```
UFD_STATUS UFD_API UFD_Close(  
    HUFDatabase hDatabase  
)
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
-----------	------------------------------------

See also

[UFD_Open](#)

Examples

Visual C++

```
UFD_STATUS ufd_res;  
HUFDatabase hDatabase;  
  
// Get database handle, hDatabase  
  
ufd_res = UFD_Close(hDatabase);  
if (ufd_res == UFD_OK) {  
    // UFD_Close is succeeded  
} else {  
    // UFD_Close is failed  
    // Use UFD\_GetErrorString function to show error string  
}
```

Visual Basic 6.0

```
Dim ufd_res As UFD_STATUS  
Dim hDatabase As Long  
  
' Get database handle, hDatabase  
  
ufd_res = UFD_Close(hDatabase)  
If (ufd_res = UFD_STATUS.OK) Then
```

```
' UFD_Close is succeeded
Else
    ' UFD_Close is failed
' Use UFD\_GetErrorString function to show error string
End If
```

UFD_AddData

Adds data into the specified database. If there is a database entry of same user ID with finger index with input data, the function returns [UFD_ERR_SAME_FINGER_EXIST](#).

```
UFD_STATUS UFD_API UFD_AddData(
    HUFDatabase hDatabase,
    const char* szUserID,
    int nFingerIndex,
    unsigned char* pTemplate1,
    int nTemplate1Size,
    unsigned char* pTemplate2,
    int nTemplate2Size,
    const char* szMemo
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#),
[UFD_ERR_SAME_FINGER_EXIST](#)

Parameters

hDatabase	[in] Handle of the database object
szUserID	[in] Specifies user ID Information
nFingerIndex	[in] Specifies finger index for Identifying finger
pTemplate1	[in] Specifies first fingerprint template data
nTemplate1Size	[in] Specifies the size of template
pTemplate2	[in] Specifies second fingerprint template data
nTemplate2Size	[in] Specifies the size of template
szMemo	[in] Specifies additional user data

Examples

Visual C++

```
#define MAX_USERID_SIZE 50
#define MAX_TEMPLATE_SIZE 1024
#define MAX_MEMO_SIZE 100

UFD_STATUS ufd_res;
HUFDatabase hDatabase;
char szUserID[MAX_USERID_SIZE];
int nFingerIndex;
unsigned char Template1[MAX_TEMPLATE_SIZE];
```

```

int nTemplate1Size;
unsigned char Template2[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
char szMemo[MAX_MEMO_SIZE];

// Get database handle, hDatabase

// Get user data, and save them to szUserID, nFingerIndex, Template1, nTemplate1Size,
// Template2, nTemplate2Size, szMemo

ufd_res = UFD_AddData(hDatabase, szUserID, nFingerIndex, Template1,
nTemplate1Size, Template2, nTemplate2Size, szMemo);
if(ufd_res == UFD_OK) {
    // UFD_AddData is succeeded
} else {
    // UFD_AddData is failed
    // Use UFD\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

Const MAX_USERID_SIZE As Long = 50
Const MAX_TEMPLATE_SIZE As Long = 1024
Const MAX_MEMO_SIZE As Long = 100

Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim UserID As String
Dim FingerIndex As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2Size As Long
Dim Memo As String

' Get database handle, hDatabase

' Get user data, and save them to UserID, FingerIndex, Template1, Template1Size,
Template2, Template2Size, Memo

ufd_res = UFD_AddData(hDatabase, UserID, FingerIndex, Template1(0), Template1Size,
Template2(0), Template2Size, Memo)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_AddData is succeeded
Else
    ' UFD_AddData is failed
    ' Use UFD\_GetErrorString function to show error string
End If

```

UFD_UpdateDataByUserInfo

Updates the database entry having specified user ID and finger index.

```
UFD_STATUS UFD_API UFD_UpdateDataByUserInfo(
    HUFDatabase hDatabase,
    const char* szUserID,
    int nFingerIndex,
    unsigned char* pTemplate1,
    int nTemplate1Size,
    unsigned char* pTemplate2,
    int nTemplate2Size,
    const char* szMemo
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
szUserID	[in] Specifies user ID Information
nFingerIndex	[in] Specifies finger index for Identifying finger
pTemplate1	[in] Specifies first fingerprint template data to update; NULL indicates no update for template1
nTemplate1Size	[in] Specifies the size of template; 0 indicates no update for template1
pTemplate2	[in] Specifies second fingerprint template data to update; NULL indicates no update for template2
nTemplate2Size	[in] Specifies the size of template; 0 indicates no update for template2
szMemo	[in] Specifies additional user data to update; NULL indicates no update for memo

See also

[UFD_UpdateDataBySerial](#)

Examples

Visual C++
#define MAX_USERID_SIZE 50 #define MAX_TEMPLATE_SIZE 1024 #define MAX_MEMO_SIZE 100

```

UFD_STATUS ufd_res;
HUFDatabase hDatabase;
char szUserID[MAX_USERID_SIZE];
int nFingerIndex;
unsigned char Template1[MAX_TEMPLATE_SIZE];
int nTemplate1Size;
unsigned char Template2[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
char szMemo[MAX_MEMO_SIZE];

// Get database handle, hDatabase

// Get user data, and save them to szUserID, nFingerIndex, Template1, nTemplate1Size,
// Template2, nTemplate2Size, szMemo

ufd_res = UFD_UpdateDataByUserInfo(hDatabase, szUserID, nFingerIndex, Template1,
nTemplate1Size, Template2, nTemplate2Size, szMemo);
if(ufd_res == UFD_OK) {
    // UFD_UpdateDataByUserInfo is succeeded
} else {
    // UFD_UpdateDataByUserInfo is failed
    // Use UFD\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

Const MAX_USERID_SIZE As Long = 50
Const MAX_TEMPLATE_SIZE As Long = 1024
Const MAX_MEMO_SIZE As Long = 100

Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim UserID As String
Dim FingerIndex As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2Size As Long
Dim Memo As String

' Get database handle, hDatabase

' Get user data, and save them to UserID, FingerIndex, Template1, Template1Size,
Template2, Template2Size, Memo

ufd_res = UFD_UpdateDataByUserInfo(hDatabase, UserID, FingerIndex, Template1(0),
Template1Size, Template2(0), Template2Size, Memo)
If(ufd_res = UFD_STATUS.OK) Then
    ' UFD_UpdateDataByUserInfo is succeeded
Else

```

```
' UFD_UpdateDataByUserInfo is failed  
' Use UFD\_GetErrorString function to show error string  
End If
```

UFD_UpdateDataBySerial

Updates the database entry having specified serial number.

```
UFD_STATUS UFD_API UFD_UpdateDataBySerial(
    HUFDatabase hDatabase,
    int nSerial,
    unsigned char* pTemplate1,
    int nTemplate1Size,
    unsigned char* pTemplate2,
    int nTemplate2Size,
    const char* szMemo
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
nSerial	[in] Specifies serial number
pTemplate1	[in] Specifies first fingerprint template data to update; NULL indicates no update for template1
nTemplate1Size	[in] Specifies the size of template; 0 indicates no update for template1
pTemplate2	[in] Specifies second fingerprint template data to update; NULL indicates no update for template2
nTemplate2Size	[in] Specifies the size of template; 0 indicates no update for template2
szMemo	[in] Specifies additional user data to update; NULL indicates no update for memo

See also

[UFD_UpdateDataByUserInfo](#)

Examples

Visual C++
<pre>#define MAX_TEMPLATE_SIZE 1024 #define MAX_MEMO_SIZE 100 UFD_STATUS ufd_res; HUFDatabase hDatabase;</pre>

```
int nSerial;
unsigned char Template1[MAX_TEMPLATE_SIZE];
int nTemplate1Size;
unsigned char Template2[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
char szMemo[MAX_Memo_SIZE];

// Get database handle, hDatabase

// Get user data, and save them to nSerial, Template1, nTemplate1Size, Template2,
nTemplate2Size, szMemo

ufd_res = UFD_UpdateDataBySerial(hDatabase, nSerial, Template1, nTemplate1Size,
Template2, nTemplate2Size, szMemo);
if(ufd_res == UFD_OK) {
    // UFD_UpdateDataBySerial is succeeded
} else {
    // UFD_UpdateDataBySerial is failed
    // Use UFD\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Const MAX_TEMPLATE_SIZE As Long = 1024
Const MAX_Memo_SIZE As Long = 100

Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim Serial As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2Size As Long
Dim Memo As String

' Get database handle, hDatabase

' Get user data, and save them to Serial, Template1, Template1Size, Template2,
Template2Size, Memo

ufd_res = UFD_UpdateDataBySerial(hDatabase, Serial, Template1(0), Template1Size,
Template2(0), Template2Size, Memo)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_UpdateDataBySerial is succeeded
Else
    ' UFD_UpdateDataBySerial is failed
    ' Use UFD\_GetErrorString function to show error string
End If
```

UFD_RemoveDataByUserID

Removes the database entries having specified user ID.

```
UFD_STATUS UFD_API UFD_RemoveDataByUserID(
    HUFDatabase hDatabase,
    const char* szUserID
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
szUserID	[in] Specifies user ID Information

See also

[UFD_RemoveDataByUserInfo](#), [UFD_RemoveDataBySerial](#), [UFD_RemoveAllData](#)

Examples

Visual C++

```
#define MAX_USERID_SIZE 50

UFD_STATUS ufd_res;
HUFDatabase hDatabase;
char szUserID[MAX_USERID_SIZE];

// Get database handle, hDatabase

// Get szUserID from user

ufd_res = UFD_RemoveDataByUserID(hDatabase, szUserID);
if(ufd_res == UFD_OK) {
    // UFD_RemoveDataByUserID is succeeded
} else {
    // UFD_RemoveDataByUserID is failed
    // Use UFD\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Const MAX_USERID_SIZE As Long = 50
```

```
Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim UserID As String

' Get database handle, hDatabase

' Get UserID from user

ufd_res = UFD_RemoveDataByUserID(hDatabase, UserID)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_RemoveDataByUserID is succeeded
Else
    ' UFD_RemoveDataByUserID is failed
    ' Use UFD_GetErrorString function to show error string
End If
```

UFD_RemoveDataByUserInfo

Removes the database entries having specified user ID and finger index.

```
UFD_STATUS UFD_API UFD_RemoveDataByUserInfo(
    HUFDatabase hDatabase,
    const char* szUserID,
    int nFingerIndex
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
szUserID	[in] Specifies user ID Information
nFingerIndex	[in] Specifies finger index for Identifying finger

See also

[UFD_RemoveDataByUserID](#), [UFD_RemoveDataBySerial](#), [UFD_RemoveAllData](#)

Examples

Visual C++

```
#define MAX_USERID_SIZE 50

UFD_STATUS ufd_res;
HUFDatabase hDatabase;
char szUserID[MAX_USERID_SIZE];
int nFingerIndex;

// Get database handle, hDatabase

// Get szUserID, nFingerIndex from user

ufd_res = UFD_RemoveDataByUserInfo(hDatabase, szUserID, nFingerIndex);
if(ufd_res == UFD_OK) {
    // UFD_RemoveDataByUserInfo is succeeded
} else {
    // UFD_RemoveDataByUserInfo is failed
    // Use UFD\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Const MAX_USERID_SIZE As Long = 50

Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim UserID As String
Dim FingerIndex As Long

' Get database handle, hDatabase

' Get UserID, FingerIndex from user

ufd_res = UFD_RemoveDataByUserInfo(hDatabase, UserID, FingerIndex)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_RemoveDataByUserInfo is succeeded
Else
    ' UFD_RemoveDataByUserInfo is failed
    ' Use UFD\_GetErrorString function to show error string
End If
```

UFD_RemoveDataBySerial

Removes the database entries having specified serial number.

```
UFD_STATUS UFD_API UFD_RemoveDataBySerial(
    HUFDatabase hDatabase,
    int nSerial
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
nSerial	[in] Specifies serial number

See also

[UFD_RemoveDataByUserID](#), [UFD_RemoveDataByUserInfo](#), [UFD_RemoveAllData](#)

Examples

Visual C++

```
UFD_STATUS ufd_res;
HUFDatabase hDatabase;
int nSerial;

// Get database handle, hDatabase

// Get nSerial from user

ufd_res = UFD_RemoveDataBySerial(hDatabase, nSerial);
if(ufd_res == UFD_OK) {
    // UFD_RemoveDataBySerial is succeeded
} else {
    // UFD_RemoveDataBySerial is failed
    // Use UFD\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
```

```
Dim Serial As Long

' Get database handle, hDatabase

' Get Serial from user

ufd_res = UFD_RemoveDataBySerial(hDatabase, Serial)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_RemoveDataBySerial is succeeded
Else
    ' UFD_RemoveDataBySerial is failed
    ' Use UFD_GetErrorString function to show error string
End If
```

UFD_RemoveAllData

Removes all database entries.

```
UFD_STATUS UFD_API UFD_UFD_RemoveAllData(
    HUFDatabase hDatabase
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
-----------	------------------------------------

See also

[UFD_RemoveDataByUserID](#), [UFD_RemoveDataByUserInfo](#), [UFD_RemoveDataBySerial](#)

Examples

Visual C++

```
UFD_STATUS ufd_res;
HUFDatabase hDatabase;

// Get database handle, hDatabase

ufd_res = UFD_RemoveAllData(hDatabase);
if(ufd_res == UFD_OK) {
    // UFD_RemoveAllData is succeeded
} else {
    // UFD_RemoveAllData is failed
    // Use UFD\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufd_res As UFD_STATUS
Dim hDatabase As Long

' Get database handle, hDatabase

ufd_res = UFD_RemoveAllData(hDatabase)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_RemoveAllData is succeeded
```

```
Else
    ' UFD_RemoveAllData is failed
    ' Use UFD\_GetErrorString function to show error string
End If
```

UFD_GetDataNumber

Gets the number of database entries.

```
UFD_STATUS UFD_API UFD_GetDataNumber(
    HUFDatabase hDatabase,
    int* pnDataNumber
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
pnDataNumber	[out] Receives the number of database entries

See also

[UFD_GetDataByIndex](#), [UFD_GetDataByUserInfo](#), [UFD_GetDataBySerial](#)

Examples

Visual C++
<pre>UFD_STATUS ufd_res; HUFDatabase hDatabase; int nDataNumber; // Get database handle, hDatabase ufd_res = UFD_GetDataNumber(hDatabase, &nDataNumber); if(ufd_res == UFD_OK) { // UFD_GetDataNumber is succeeded } else { // UFD_GetDataNumber is failed // Use UFD_GetErrorString function to show error string }</pre>

Visual Basic 6.0
<pre>Dim ufd_res As UFD_STATUS Dim hDatabase As Long Dim DataNumber As Long ' Get database handle, hDatabase</pre>

```
ufd_res = UFD_GetDataNumber(hDatabase, DataNumber)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_GetDataNumber is succeeded
Else
    ' UFD_GetDataNumber is failed
    ' Use UFD\_GetErrorString function to show error string
End If
```

UFD_GetDataByIndex

Gets the database entry having specified index.

```
UFD_STATUS UFD_API UFD_GetDataByIndex(
    HUFDatabase hDatabase,
    int nIndex,
    int* pnSerial,
    char* szUserID,
    int* pnFingerIndex,
    unsigned char* pTemplate1,
    int* pnTemplate1Size,
    unsigned char* pTemplate2,
    int* pnTemplate2Size,
    char* szMemo
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
nIndex	[in] Specifies the index
pnSerial	[out] Receives the serial number
szUserID	[out] Receives user ID Information
pnFingerIndex	[out] Receives finger index
pTemplate1	[out] Receives first fingerprint template data
pnTemplate1Size	[out] Receives the size of template
pTemplate2	[out] Receives second fingerprint template data
pnTemplate2Size	[out] Receives the size of template
szMemo	[out] Receives additional user data

See also

[UFD_GetDataNumber](#), [UFD_GetDataByUserInfo](#), [UFD_GetDataBySerial](#)

Examples

Visual C++
#define MAX_USERID_SIZE 50

```
#define MAX_TEMPLATE_SIZE 1024
#define MAX_Memo_SIZE 100

UFD_STATUS ufd_res;
HUFDatabase hDatabase;
int nIndex;
int nSerial;
char szUserID[MAX_USERID_SIZE];
int nFingerIndex;
unsigned char Template1[MAX_TEMPLATE_SIZE];
int nTemplate1Size;
unsigned char Template2[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
char szMemo[MAX_Memo_SIZE];

// Get database handle, hDatabase

// Get nIndex from user

ufd_res = UFD_GetDataByIndex(hDatabase, nIndex, &nSerial, szUserID,
&nFingerIndex, Template1, &nTemplate1Size, Template2, &nTemplate2Size, szMemo);
if (ufd_res == UFD_OK) {
    // UFD_GetDataByIndex is succeeded
} else {
    // UFD_GetDataByIndex is failed
    // Use UFD\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Const MAX_USERID_SIZE As Long = 50
Const MAX_TEMPLATE_SIZE As Long = 1024
Const MAX_Memo_SIZE As Long = 100

Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim Index As Long
Dim Serial As Long
Dim UserID As String
Dim FingerIndex As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2Size As Long
Dim Memo As String

' Get database handle, hDatabase

' Get Index from user
```

```
ufd_res = UFD_GetDataByIndex(hDatabase, Index, Serial, UserID, FingerIndex,
Template1(0), Template1Size, Template2(0), Template2Size, Memo)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_GetDataByIndex is succeeded
Else
    ' UFD_GetDataByIndex is failed
    ' Use UFD\_GetErrorString function to show error string
End If
```

UFD_GetDataByUserInfo

Gets the database entry having specified user ID and finger index.

```
UFD_STATUS UFD_API UFD_GetDataByUserInfo(
    HUFDatabase hDatabase,
    const char* szUserID,
    int nFingerIndex,
    unsigned char* pTemplate1,
    int* pnTemplate1Size,
    unsigned char* pTemplate2,
    int* pnTemplate2Size,
    char* szMemo
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
szUserID	[in] Specifies user ID Information
pnFingerIndex	[in] Specifies finger index
pTemplate1	[out] Receives first fingerprint template data
pnTemplate1Size	[out] Receives the size of template
pTemplate2	[out] Receives second fingerprint template data
pnTemplate2Size	[out] Receives the size of template
szMemo	[out] Receives additional user data

See also

[UFD_GetDataNumber](#), [UFD_GetDataByIndex](#), [UFD_GetDataBySerial](#)

Examples

Visual C++

```
#define MAX_USERID_SIZE 50
#define MAX_TEMPLATE_SIZE 1024
#define MAX_MEMO_SIZE 100

UFD_STATUS ufd_res;
HUFDatabase hDatabase;
char szUserID[MAX_USERID_SIZE];
```

```

int nFingerIndex;
unsigned char Template1[MAX_TEMPLATE_SIZE];
int nTemplate1Size;
unsigned char Template2[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
char szMemo[MAX_Memo_SIZE];

// Get database handle, hDatabase

// Get szUserID, nFingerIndex from user

ufd_res = UFD_GetDataByUserInfo(hDatabase, szUserID, nFingerIndex, Template1,
&nTemplate1Size, Template2, &nTemplate2Size, szMemo);
if (ufd_res == UFD_OK) {
    // UFD_GetDataByUserInfo is succeeded
} else {
    // UFD_GetDataByUserInfo is failed
    // Use UFD\_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

Const MAX_USERID_SIZE As Long = 50
Const MAX_TEMPLATE_SIZE As Long = 1024
Const MAX_Memo_SIZE As Long = 100

Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim UserID As String
Dim FingerIndex As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2Size As Long
Dim Memo As String

' Get database handle, hDatabase

' Get UserID, FingerIndex from user

ufd_res = UFD_GetDataByUserInfo(hDatabase, UserID, FingerIndex, Template1(0),
Template1Size, Template2(0), Template2Size, Memo)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_GetDataByUserInfo is succeeded
Else
    ' UFD_GetDataByUserInfo is failed
    ' Use UFD\_GetErrorString function to show error string
End If

```

UFD_GetDataBySerial

Gets the database entry having specified serial number.

```
UFD_STATUS UFD_API UFD_GetDataBySerial(
    HUFDatabase hDatabase,
    int nSerial,
    const char* szUserID,
    int* pnFingerIndex,
    unsigned char* pTemplate1,
    int* pnTemplate1Size,
    unsigned char* pTemplate2,
    int* pnTemplate2Size,
    char* szMemo
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
nSerial	[in] Specifies the serial number
szUserID	[out] Receives user ID Information
pnFingerIndex	[out] Receives finger index
pTemplate1	[out] Receives first fingerprint template data
pnTemplate1Size	[out] Receives the size of template
pTemplate2	[out] Receives second fingerprint template data
pnTemplate2Size	[out] Receives the size of template
szMemo	[out] Receives additional user data

See also

[UFD_GetDataNumber](#), [UFD_GetDataByIndex](#), [UFD_GetDataByUserInfo](#)

Examples

Visual C++

```
#define MAX_USERID_SIZE 50
#define MAX_TEMPLATE_SIZE 1024
#define MAX_MEMO_SIZE 100

UFD_STATUS ufd_res;
```

```

HUFDatabase hDatabase;
int nSerial;
char szUserID[MAX_USERID_SIZE];
int nFingerIndex;
unsigned char Template1[MAX_TEMPLATE_SIZE];
int nTemplate1Size;
unsigned char Template2[MAX_TEMPLATE_SIZE];
int nTemplate2Size;
char szMemo[MAX_MEMO_SIZE];

// Get database handle, hDatabase

// Get nSerial from user

ufd_res = UFD_GetDataBySerial(hDatabase, Serial, szUserID, &nFingerIndex,
Template1, &nTemplate1Size, Template2, &nTemplate2Size, szMemo);
if (ufd_res == UFD_OK) {
    // UFD_GetDataBySerial is succeeded
} else {
    // UFD_GetDataBySerial is failed
    // Use UFD_GetErrorString function to show error string
}

```

Visual Basic 6.0

```

Const MAX_USERID_SIZE As Long = 50
Const MAX_TEMPLATE_SIZE As Long = 1024
Const MAX_MEMO_SIZE As Long = 100

Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim Serial As Long
Dim UserID As String
Dim FingerIndex As Long
Dim Template1(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template1Size As Long
Dim Template2(MAX_TEMPLATE_SIZE - 1) As Byte
Dim Template2Size As Long
Dim Memo As String

' Get database handle, hDatabase

' Get Serial from user

ufd_res = UFD_GetDataBySerial(hDatabase, Serial, UserID, FingerIndex, Template1(0),
Template1Size, Template2(0), Template2Size, Memo)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_GetDataBySerial is succeeded
Else
    ' UFD_GetDataBySerial is failed

```

```
' Use UFD\_GetErrorString function to show error string  
End If
```

UFD_GetTemplateNumber

Gets the number of templates in specified database.

```
UFD_STATUS UFD_API UFD_GetTemplateNumber(
    HUFDatabase hDatabase,
    int* pnTemplateNumber
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
pnTemplateNumber	[out] Receives the number of templates in the database

See also

[UFD_GetTemplateListWithSerial](#)

Examples

Visual C++

```
UFD_STATUS ufd_res;
HUFDatabase hDatabase;
int nTemplateNumber;

// Get database handle, hDatabase

ufd_res = UFD_GetTemplateNumber(hDatabase, &nTemplateNumber);
if(ufd_res == UFD_OK) {
    // UFD_GetTemplateNumber is succeeded
} else {
    // UFD_GetTemplateNumber is failed
    // Use UFD\_GetErrorString function to show error string
}
```

Visual Basic 6.0

```
Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim TemplateNumber As Long
```

```
' Get database handle, hDatabase  
  
ufd_res = UFD_GetTemplateNumber(hDatabase, TemplateNumber)  
If (ufd_res = UFD_STATUS.OK) Then  
    ' UFD_GetTemplateNumber is succeeded  
Else  
    ' UFD_GetTemplateNumber is failed  
    ' Use UFD\_GetErrorString function to show error string  
End If
```

UFD_GetTemplateListWithSerial

Gets all the template list with corresponding serial number list from specified database.

```
UFD_STATUS UFD_GetTemplateListWithSerial(
    HUFDatabase hDatabase,
    unsigned char** ppTemplate,
    int* pnTemplateSize,
    int* pnSerial
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

hDatabase	[in] Handle of the database object
ppTemplate	[out] Receives the template list
pnTemplateSize	[out] Receives the template size list
pnSerial	[out] Receives the serial number list

See also

[UFD_GetTemplateName](#)

Examples

Visual C++
<pre>#define MAX_TEMPLATE_SIZE 1024 UFD_STATUS ufd_res; HUFDatabase hDatabase; int nTemplateNumber; unsigned char** ppTemplate; int* pnTemplateSize; int* pnSerial; int i; // Get database handle, hDatabase // Get nTemplateNumber using UFD_GetTemplateName function // Allocate templates list and serial list ppTemplate = (unsigned char**)malloc(nTemplateNumber * sizeof(unsigned char*));</pre>

```

for (i = 0; i < nTemplate2Num; i++) {
    ppTemplate[i] = (unsigned char*)malloc(MAX_TEMPLATE_SIZE *
    sizeof(unsigned char));
}
pnTemplateSize = (int*)malloc(nTemplateNumber * sizeof(int));
pnSerial = (int*)malloc(nTemplateNumber * sizeof(int));

ufd_res = UFD_GetTemplateListWithSerial(hDatabase, ppTemplate, pnTemplateSize,
pnSerial);
if (ufd_res == UFD_OK) {
    // UFD_GetTemplateListWithSerial is succeeded
} else {
    // UFD_GetTemplateListWithSerial is failed
    // Use UFD\_GetErrorString function to show error string
}

// Use template list and serial list

// Free templates list
for (i = 0; i < nTemplateNumber; i++) {
    free(ppTemplate[i]);
}
free(ppTemplate);
free(pnTemplateSize);
free(pnSerial);

```

Visual Basic 6.0

```

Const MAX_TEMPLATE_SIZE As Long = 1024

Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim TemplateNumber As Long
Dim Template() As Byte
Dim TemplateSize() As Long
Dim TemplatePtr() As Long
Dim Serial() As Long

' Get database handle, hDatabase

' Get TemplateNumber using UFD\_GetTemplateNumber function

' Allocate templates list and serial list
ReDim Template(MAX_TEMPLATE_SIZE - 1, TemplateNumber - 1) As Byte
ReDim TemplateSize(TemplateNumber - 1) As Long
ReDim Serial(TemplateNumber - 1) As Long

' Make template pointer array to pass two dimensional template data
ReDim TemplatePtr(TemplateNumber - 1) As Long
For i = 0 To TemplateNumber - 1

```

```
TemplatePtr(i) = VarPtr(Template2(0, i))
Next

ufd_res = UFD_GetTemplateListWithSerial(hDatabase, TemplatePtr(0), TemplateSize(0),
Serial(0))
If (ufd_res = UFD_STATUS.OK) Then
    'UFD_GetDataBySerial is succeeded
Else
    'UFD_GetDataBySerial is failed
    'Use UFD\_GetErrorString function to show error string
End If
```

UFD_GetErrorString

Gets the error string for specified [UFD_STATUS](#) value.

```
UFD\_STATUS UFD_GetErrorString(
    UFD\_STATUS res,
    char* szErrorString
);
```

Possible return values

[UFD_OK](#), [UFD_ERROR](#), [UFD_ERR_INVALID_PARAMETERS](#)

Parameters

res	[in] Status return value
szErrorString	[out] Receives error sting

Examples

Visual C++

```
UFD_STATUS ufd_res;
char strError[128];

// Get status return code, ufd_res

ufd_res = UFD_GetErrorString(ufd_res, strError);
if(ufd_res == UFD_OK) {
    // UFD_GetErrorString is succeeded
} else {
    // UFD_GetErrorString is failed
}
```

Visual Basic 6.0

```
Dim ufd_res As UFD_STATUS
Dim m_strError As String

' Get status return code, ufd_res

ufd_res = UFD_GetErrorString(ufd_res, strError)
If (ufd_res = UFD_STATUS.OK) Then
    ' UFD_GetErrorString is succeeded
Else
    ' UFD_GetErrorString is failed
End If
```

Chapter 6. Reference (.NET)

This chapter contains reference of all modules included in Suprema PC SDK for .NET developers using following languages,

- Visual C#
- Visual Basic .NET

In this chapter, APIs are described using C# language.

Suprema.UFScanner module

Suprema.UFScanner module provides functionality for managing scanners, capture finger images from scanners, extracting templates from captured images using scanners, etc. Actually, Suprema.UFScanner module is C# wrapper dll of UFScanner.dll. The module is created using Visual C# 7.1 and compatible with .NET Framework 1.1 / 2.0 or higher.

Requirements

Visual C#, Visual Basic .NET

- Required reference: bin\Suprema.UFScanner.dll
- Required dll: bin\Suprema.UFScanner.dll, bin\UFScanner.dll

Supported scanners

List of supported scanners could be found in [UFScanner module](#).

Suprema

Suprema namespace

Enumerations

UFS_STATUS	Every method in UFScanner module returns UFS_STATUS enumeration value
UFS_SCANNER_TYPE	UFScanner.ScannerType property gives UFS_SCANNER_TYPE enumeration value

Delegates

UFS_SCANNER_PROC	Defines the delegate for the scanner event ScannerEvent
UFS_CAPTURE_PROC	Defines the delegate for the capture event CaptureEvent

Classes

UFScannerManagerScannerEventArgs	Contains data of event ScannerEvent of UFScannerManager class
UFScannerCaptureEventArgs	Contains data of event CaptureEvent of UFScanner class
UFScannerManager	Provides functionality for managing scanners
UFScannerManager.ScannerList	Holds UFScanner list of the connected scanners
UFScanner	Provides functionality for capturing finger images from scanners, extracting templates from captured images using scanners, etc

UFS_STATUS enumeration

Every function in UFScanner module returns UFS_STATUS enumeration value.

public enum UFS_STATUS : int

Members

UFS_STATUS member name	Code	Meaning
OK	0	Success
ERROR	-1	General error
ERR_NO_LICENSE	-101	System has no license
ERR_LICENSE_NOT_MATCH	-102	License is not match
ERR_LICENSE_EXPIRED	-103	License is expired
ERR_NOT_SUPPORTED	-111	This function is not supported
ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
ERR_ALREADY_INITIALIZED	-201	Module is already initialized
ERR_NOT_INITIALIZED	-202	Module is not initialized
ERR_DEVICE_NUMBER_EXCEED	-203	Device number is exceed
ERR_LOAD_SCANNER_LIBRARY	-204	Error on loading scanner library
ERR_CAPTURE_RUNNING	-211	Capturing is started using UFScanner.CaptureSingleImage or UFScanner.StartCapturing
ERR_CAPTURE_FAILED	-212	Capturing is timeout or aborted
ERR_NOT_GOOD_IMAGE	-301	Input image is not good
ERR_EXTRACTION_FAILED	-302	Extraction is failed
ERR_CORE_NOT_DETECTED	-351	Core is not detected
ERR_CORE_TO_LEFT	-352	Move finger to left
ERR_CORE_TO_LEFT_TOP	-353	Move finger to left-top
ERR_CORE_TO_TOP	-354	Move finger to top
ERR_CORE_TO_RIGHT_TOP	-355	Move finger to right-top
ERR_CORE_TO_RIGHT	-356	Move finger to right
ERR_CORE_TO_RIGHT_BOTTOM	-357	Move finger to right-bottom
ERR_CORE_TO_BOTTOM	-358	Move finger to bottom
ERR_CORE_TO_LEFT_BOTTOM	-359	Move finger to left-bottom

UFS_SCANNER_TYPE enumeration

[UFScanner.ScannerType](#) property gives UFS_SCANNER_TYPE enumeration value.

```
public enum UFS_SCANNER_TYPE : int
```

Members

UFS_SCANNER_TYPE member name	Code	Meaning
SFR200	1001	Suprema SFR200
SFR300	1002	Suprema SFR300-S
SFR300v2	1003	Suprema SFR300-S(Ver.2)

UFS_SCANNER_PROC delegate

Defines the delegate for the scanner event [ScannerEvent](#)

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

Parameters

sender	The sender of the event
e	A UFScannerManagerScannerEventArgs that contains the event data

UFS_CAPTURE_PROC delegate

Defines the delegate for the capture event [CaptureEvent](#)

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

Parameters

sender	The sender of the event
e	A UFScannerCaptureEventArgs that contains the event data

UFScannerManagerScannerEventArgs class

Contains data of event [ScannerEvent](#) of [UFScannerManager](#) class.

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

Properties

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

UFScannerCaptureEventArgs class

Contains data of event [CaptureEvent](#) of [UFScanner](#) class.

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

Properties

ImageFrame	Receives a captured image
Resolution	Receives the resolution of ImageFrame
FingerOn	true: finger is on the scanner, false: finger is not on the scanner

UFScannerManager class

Constructors

UFScannerManager	Initializes a new instance of the UFScannerManager class
----------------------------------	--

Properties

Scanners	Gets connected scanners as UFScannerManager.ScannerList
--------------------------	---

Events

ScannerEvent	Occurs when the scanner is connected or disconnected
------------------------------	--

Methods

Init	Initializes scanner module
Update	Enforces scanner module to update current connectivity of scanners
Uninit	Un-initializes scanner module
GetScannerNumber	Gets the number of scanners

UFScannerManager constructor

Initializes a new instance of the [UFScannerManager](#) class.

```
public UFScannerManager(  
    ISynchronizeInvoke synInvoke  
>;
```

Parameters

synInvoke	An ISynchronizeInvoke object
-----------	------------------------------

Scanners property

Gets connected scanners as [UFScannerManager.ScannerList](#).

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

ScannerEvent event

Occurs when the scanner is connected or disconnected. Scanner event is not working for every scanner model. Currently this functionality is working for Suprema SFR300-S(Ver.2) in windows 2000 / 2003 / XP only.

```
public event UFS_SCANNER_PROC ScannerEvent;
```

Init method

Initializes scanner module.

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_ALREADY_INITIALIZED](#), [UFS_STATUS.ERR_NO_LICENSE](#),
[UFS_STATUS.ERR_LICENSE_NOT_MATCH](#),
[UFS_STATUS.ERR_LICENSE_EXPIRED](#),
[UFS_STATUS.ERR_DEVICE_NUMBER_EXCEED](#)

Update method

Enforces scanner module to update current connectivity of scanners.

```
public UFS\_STATUS Update();
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#), [UFS_STATUS.ERR_NOT_INITIALIZED](#),
[UFS_STATUS.ERR_NO_LICENSE](#), [UFS_STATUS.ERR_LICENSE_NOT_MATCH](#),
[UFS_STATUS.ERR_LICENSE_EXPIRED](#),
[UFS_STATUS.ERR_DEVICE_NUMBER_EXCEED](#)

Uninit method

Un-initializes scanner module.

```
public UFS\_STATUS Uninit();
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#), [UFS_STATUS.ERR_NOT_INITIALIZED](#),
[UFS_STATUS.ERR_NO_LICENSE](#), [UFS_STATUS.ERR_LICENSE_NOT_MATCH](#),
[UFS_STATUS.ERR_LICENSE_EXPIRED](#)

UFScannerManager.ScannerList class

Properties

Count	Gets the number of connected scanners
Item	Gets UFScanner reference by index or handle or ID of scanner

Count property

Gets the number of connected scanners.

```
public int Count { get; }
```

Item property

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; }
```

UFScanner class

Events

CaptureEvent	Occurs when an image frame is captured from the scanner
------------------------------	---

Properties

ID	Gets scanner ID assigned to the scanner
Timeout	Gets or sets timeout value of the scanner
Brightness	Gets or sets brightness value of the scanner
Sensitivity	Gets or sets sensitivity value of the scanner
Serial	Gets scanner serial assigned to the scanner
DetectFake	Gets or sets Fake Detection value of the scanner.
DetectCore	Gets or sets whether detecting core when extracting templates
TemplateSize	Gets or sets template size limitation of the scanner
UseSIF	Gets or sets whether using SIF for templates
ScannerType	Gets scanner type of the scanner
IsSensorOn	Checks the scanner is connected or not
IsFingerOn	Checks a finger in placed on the scanner or not
IsCapturing	Checks capture process is running
Handle	Gets handle assigned to the scanner

Methods

SetScanner	Sets current scanner instance using index / handle / ID information
CaptureSingleImage	Captures single image. Captured image is stored to the internal buffer
StartCapturing	Starts capturing
AbortCapturing	Aborts capturing
Extract	Extracts a template from the stored image buffer
SetEncryptionKey	Sets encryption key
EncryptTemplate	Encrypts template
DecryptTemplate	Decrypts template
GetCaptureImageBuffer	Gets captured image from the buffer
DrawCaptureImageBuffer	Draws the fingerprint image
DrawFeatureImageBuffer	Draws the features on fingerprint image
SaveCaptureImageBufferToBMP	Saves the capture image buffer to the specified file of the bitmap format
SaveCaptureImageBufferToTIF	Saves the capture image buffer to the specified file of the tiff format

<u>SaveCaptureImageBufferToJPG</u>	Saves the capture image buffer to the specified file of the jpeg format
<u>ClearCaptureImageBuffer</u>	Clears the capture image buffer
<u>SelectTemplate</u>	Selects good templates from input templates
<u>GetErrorString</u>	Gets the error string

CaptureEvent event

After a capturing is started using [StartCapturing](#), this event occurs when an image frame is captured from the scanner.

```
public event UFS_CAPTURE_PROC CaptureEvent;
```

ID property

Gets scanner ID assigned to the scanner.

```
public string ID { get; }
```

Timeout property

Gets or sets timeout value of the scanner. The unit is millisecond and 0 means infinity. Default value is 5000.

```
public int Timeout { get; set; }
```

Brightness property

Gets or sets brightness value of the scanner. The value ranges from 0 to 255. Higher value means darker image. Default value is 100.

```
public int Brightness { get; set; }
```

Sensitivity property

Gets or sets sensitivity value of the scanner. The value ranges from 0 to 7. The higher value is, the more sensitive. Default value is 4.

```
public int Sensitivity { get; set; }
```

DetectFake property

Gets or sets Fake Detection value of the scanner. The value ranges from 0 to 3. The higher value is, the more strong filter. Default value 0 means not using LFD.

```
public int DetectFake { get; set; }
```

Serial property

Gets scanner serial assigned to the scanner.

```
public string Serial { get; }
```

DetectCore property

Gets or sets whether detecting core when extracting templates. Default value is false.

```
public bool DetectCore { get; set; }
```

TemplateSize property

Gets or sets template size limitation of the scanner. The unit is bytes and the value ranges from 256 to 1024 and step size is 32. Default value is 384.

```
public int TemplateSize { get; set; }
```

UseSIF property

Gets or sets whether using SIF (biometric data standard interchange format) for templates.
Default value is false.

```
public bool UseSIF { get; set; }
```

ScannerType property

Gets scanner type of the scanner.

```
public UFS\_SCANNER\_TYPE ScannerType { get; }
```

IsSensorOn property

Checks the scanner is connected or not.

```
public bool IsSensorOn { get; }
```

IsFingerOn property

Checks a finger in placed on the scanner or not.

```
public bool IsSensorOn { get; }
```

IsCapturing property

Checks if the specified scanner is running capturing which is started by [CaptureSingleImage](#) or [StartCapturing](#).

```
public bool IsCapturing { get; }
```

See also

[CaptureSingleImage](#), [StartCapturing](#), [AbortCapturing](#)

Handle property

Gets handle assigned to the scanner.

```
public IntPtr Handle { get; }
```

SetScanner method

Sets current scanner instance using index / handle / ID information.

```
public UFS\_STATUS SetScanner(  
    int ScannerIndex  
);  
public UFS\_STATUS SetScanner(  
    IntPtr ScannerHandle  
);  
public UFS\_STATUS SetScanner(  
    string ScannerID  
);
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#)

Parameters

ScannerIndex	[in] Set scanner instance using scanner index
ScannerHandle	[in] Set scanner instance using scanner handle
ScannerID	[in] Set scanner instance using scanner ID

CaptureSingleImage method

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

```
public UFS\_STATUS CaptureSingleImage();
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#),
[UFS_STATUS.ERR_CAPTURE_RUNNING](#)

See also

[IsCapturing](#), [AbortCapturing](#)

StartCapturing method

Starts capturing. Whenever an image frame is captured, [CaptureEvent](#) is raised. Currently this method is working for Suprema SFR300-S(Ver.2) only.

```
public UFS\_STATUS StartCapturing();
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#), [UFS_STATUS.ERR_NOT_SUPPORTED](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#),
[UFS_STATUS.ERR_CAPTURE_RUNNING](#)

Supported scanners

Suprema SFR300-S(Ver.2)

See also

[IsCapturing](#), [AbortCapturing](#)

AbortCapturing method

Aborts capturing which is started by [CaptureSingleImage](#) or [StartCapturing](#).

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#), [UFS_STATUS.ERR_NOT_SUPPORTED](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#),
[UFS_STATUS.ERR_CAPTURE_RUNNING](#)

See also

[CaptureSingleImage](#), [StartCapturing](#), [IsCapturing](#)

Extract method

Extracts a template from the stored image buffer which is acquired using [CaptureSingleImage\(\)](#) or [StartCapturing\(\)](#).

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_LICENSE_NOT_MATCH](#),
[UFS_STATUS.ERR_LICENSE_EXPIRED](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#),
[UFS_STATUS.ERR_NOT_GOOD_IMAGE](#),
[UFS_STATUS.ERR_EXTRACTION_FAILED](#),
[UFS_STATUS.ERR_CORE_NOT_DETECTED](#), [UFS_STATUS.ERR_CORE_TO_LEFT](#),
[UFS_STATUS.ERR_CORE_TO_LEFT_TOP](#), [UFS_STATUS.ERR_CORE_TO_TOP](#),
[UFS_STATUS.ERR_CORE_TO_RIGHT_TOP](#), [UFS_STATUS.ERR_CORE_TO_RIGHT](#),
[UFS_STATUS.ERR_CORE_TO_RIGHT_BOTTOM](#),
[UFS_STATUS.ERR_CORE_TO_BOTTOM](#),
[UFS_STATUS.ERR_CORE_TO_LEFT_BOTTOM](#)

Parameters

Template	[out] Receives the template array; The array must be allocated in advance
TemplateSize	[out] Receives the size (in bytes) of Template
EnrollQuality	[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.

SetEncryptionKey method

Sets encryption key.

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Key	[in] 32 bytes key array; default key is first byte is 1 and second to 32th byte are all 0
-----	---

EncryptTemplate method

Encrypts template.

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

TemplateInput	[in] Input template array
TemplateInputSize	[in] Input template size
TemplateOutput	[out] Output template array
TemplateOutputSize	[in / out] Inputs allocated size of output template array; Receives output template size

DecryptTemplate method

Decrypts template.

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

TemplateInput	[in] Input template array
TemplateInputSize	[in] Input template size
TemplateOutput	[out] Output template array
TemplateOutputSize	[in / out] Inputs allocated size of output template array; Receives output template size

GetCaptureImageBuffer method

Gets captured image from the buffer.

```
public UFS\_STATUS GetCaptureImageBuffer(  
    Bitmap bitmap,  
    int Resolution  
);
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

bitmap	[out] Receives a bitmap instance
Resolution	[out] Receives the resolution of bitmap

DrawCaptureImageBuffer method

Draws the fingerprint image which is acquired using [CaptureSingleImage\(\)](#) or [StartCapturing\(\)](#).

```
public UFS\_STATUS DrawCaptureImageBuffer(  
    Graphics g,  
    Rectangle rect,  
    bool DrawCore  
);
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

g	[in] Specifies graphics object to draw image buffer
rect	[in] Specifies position and size of image to be drawn
DrawCore	[in] Specifies whether the core of fingerprint is drawn or not

DrawFeatureImageBuffer method

Draws a fingerprint image which is acquired using [CaptureSingleImage\(\)](#) or [StartCapturing\(\)](#). This function 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 features in the image frame.



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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

g	[in] Specifies graphics object to draw image buffer
rect	[in] Specifies position and size of image to be drawn
DrawCore	[in] Specifies whether the core of fingerprint is drawn or not

SaveCaptureImageBufferToBMP method

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

```
public UFS\_STATUS SaveCaptureImageBufferToBMP(  
    string FileName  
);
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

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

SaveCaptureImageBufferToTIF method

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

```
public UFS_STATUS SaveCaptureImageBufferToTIF(  
    string FileName  
>;
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

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

SaveCaptureImageBufferToJPG method

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

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

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

ClearCaptureImageBuffer method

Clears the capture image buffer.

```
public UFS\_STATUS ClearCaptureImageBuffer();
```

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

GetErrorString method

Gets the error string for specified [UFS_STAUS](#) value.

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

res	[in] Status return value
ErrorString	[out] Receives error string

SaveCaptureImageBufferTo19794_4 method

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

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERROR](#),
[UFS_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

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

SelectTemplate method

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

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

Possible return values

[UFS_STATUS.OK](#), [UFS_STATUS.ERR_INVALID_PARAMETERS](#),
[UFS_STATUS.ERR_NOT_SUPPORTED](#)

Parameters

ppTemplateInput	[in] Input Template arrays
pnTemplateInputSize	[in] Size of each input Template
nTemplateInputNum	[in] Number of input Template arrays
ppTemplateOutput	[out] Output Template arrays
pnTemplateOutputSize	[out] Size of each output Template
nTemplateOutputNum	[in] Number of output Templates; should be less than input template number by more than one

Suprema.UFMatcher module

Suprema.UFMatcher module provides functionality for verifying fingerprints using two templates, identifying fingerprints using the template array, etc. Actually, Suprema.UFMatcher module is C# wrapper dll of UFMatcher.dll. The module is created using Visual C# 7.1 and compatible with .NET Framework 1.1 / 2.0 or higher.

Requirements

Visual C#, Visual Basic .NET

- Required reference: bin\Suprema.UFMatcher.dll
- Required dll: bin\Suprema.UFMatcher.dll, bin\UFMatcher.dll

Suprema

Suprema namespace

Enumerations

<u>UFM_STATUS</u>	Every function in UFMatcher module returns UFM_STATUS enumeration value
-----------------------------------	---

Classes

<u>UFMatcher</u>	UFMatcher class provides functionality for verifying fingerprints using two templates, identifying fingerprints using the template array, etc
----------------------------------	---

UFM_STATUS enumeration

Every function in UFM matcher module returns UFM_STATUS enumeration value.

```
public enum UFM_STATUS : int
```

Members

UFM_STATUS member name	Code	Meaning
OK	0	Success
ERROR	-1	General error
ERR_NO_LICENSE	-101	System has no license
ERR_LICENSE_NOT_MATCH	-102	License is not match
ERR_LICENSE_EXPIRED	-103	License is expired
ERR_NOT_SUPPORTED	-111	This function is not supported
ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
ERR_MATCH_TIMEOUT	-401	Matching is timeout
ERR_MATCH_ABORTED	-402	Matching is aborted
ERR_TEMPLATE_TYPE	-411	Template type is not match

UFMatcher class

Properties

FastMode	Gets or sets whether fast mode is enable in identify method
SecurityLevel	Gets or sets security level used in verify and identify method
UseSIF	Gets or sets whether using SIF for templates

Methods

Verify	Compares two extracted templates
Identify	Compares a template with given template array
IdentifyMT	Use multi threads internally for faster identifying in multi-core systems
AbortIdentify	Aborts current identifying procedure
IdentifyInit	Initializes identify with input template
IdentifyNext	Matches one input template to the template
RotateTemplate	Rotates the specified template to the amount of 180 degrees
GetErrorString	Gets the error string

FastMode property

Gets or sets whether fast mode is enable in identify method. Default value is true.

```
public bool FastMode { get; set; }
```

SecurityLevel property

Gets or sets security level used in verify and identify method. The value ranges from 1 to 7. Default value is 4.

```
public int SecurityLevel { get; set; }
```

Relation between security level and false accept ratio

Level	False Accept Ratio (FAR)
1	Below 1 % (1e-2)
2	Below 0.1 % (1e-3)
3	Below 0.01 % (1e-4)
4	Below 0.001 % (1e-5)
5	Below 0.0001 % (1e-6)
6	Below 0.00001 % (1e-7)
7	Below 0.000001 % (1e-8)

UseSIF property

Gets or sets whether using SIF (biometric data standard interchange format) for templates.
Default value is false.

```
public bool UseSIF { get; set; }
```

Verify method

Compares two extracted templates.

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

Possible return values

[UFM_STATUS.OK](#), [UFM_STATUS.ERROR](#),
[UFM_STATUS.ERR_LICENSE_NOT_MATCH](#),
[UFM_STATUS.ERR_LICENSE_EXPIRED](#),
[UFM_STATUS.ERR_INVALID_PARAMETERS](#),
[UFM_STATUS.ERR_TEMPLATE_TYPE](#)

Parameters

Template1	[in] Specifies first template
Template1Size	[in] Specifies the size of first template
Template2	[in] Specifies second template
Template2Size	[in] Specifies the size of second template
VerifySucceed	[out] Receives whether verification is succeed; true: verification is succeed, false: verification is failed

Identify, IdentifyMT method

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

```
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
);
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
);
```

Possible return values

[UFM STATUS.OK](#), [UFM STATUS.ERROR](#),
[UFM STATUS.ERR LICENSE NOT MATCH](#),
[UFM STATUS.ERR LICENSE EXPIRED](#),
[UFM STATUS.ERR INVALID PARAMETERS](#),
[UFM STATUS.ERR MATCH TIMEOUT](#), [UFM STATUS.ERR MATCH ABORTED](#),
[UFM STATUS.ERR TEMPLATE TYPE](#)

Parameters

Template1	[in] Specifies input template
Template1Size	[in] Specifies the size of input template
Template2Array	[in] Specifies the template array
Template2SizeArray	[in] Specifies the template size array
Template2Num	[in] Specifies the size of Template2Array or Template2SizeArray
Timeout	[in] Specifies maximum time for identifying in milliseconds; If elapsed time for identifying exceeds nTimeout, function stops further identifying and returns UFM_STATUS.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

AbortIdentify method

Aborts current identifying procedure started using [Identify\(\)](#).

```
public UFM\_STATUS AbortIdentify();
```

Possible return values

[UFM_STATUS.OK](#), [UFM_STATUS.ERROR](#)

IdentifyInit method

Initializes identify with input template.

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

Possible return values

[UFM_STATUS.OK](#), [UFM_STATUS.ERROR](#),
[UFM_STATUS.ERR_LICENSE_NOT_MATCH](#),
[UFM_STATUS.ERR_LICENSE_EXPIRED](#),
[UFM_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Template1	[in] Specifies input template
Template1Size	[in] Specifies the size of input template

IdentifyNext method

Matches one input template to the template specified in [IdentifyInit\(\)](#).

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

Possible return values

[UFM_STATUS.OK](#), [UFM_STATUS.ERROR](#),
[UFM_STATUS.ERR_LICENSE_NOT_MATCH](#),
[UFM_STATUS.ERR_LICENSE_EXPIRED](#),
[UFM_STATUS.ERR_INVALID_PARAMETERS](#),
[UFM_STATUS.ERR_TEMPLATE_TYPE](#)

Parameters

Template2	[in] Specifies the template
Template2Size	[in] Specifies the size of the template
IdentifySucceed	[out] Receives whether identification is succeed; true: identification is succeed, false: identification is failed

RotateTemplate method

Rotates the specified template to the amount of 180 degrees.

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

Possible return values

[UFM_STATUS.OK](#), [UFM_STATUS.ERROR](#),
[UFM_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Template	[in / out] The template
TemplateSize	[in] Specifies the size of the template

GetErrorString method

Gets the error string for specified [UFM_STATUS](#) value.

```
public UFM_STATUS GetErrorString(  
    UFM_STATUS res,  
    out string ErrorString  
>;
```

Possible return values

[UFM_STATUS.OK](#), [UFM_STATUS.ERROR](#),
[UFM_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

res	[in] Status return value
szErrorString	[out] Receives error sting

Suprema.UFExtractor module

Suprema.UFExtractor module provides functionality for extracting templates from input images, etc. Actually, Suprema.UFExtractor module is C# wrapper dll of UFExtractor.dll. The module is created using Visual C# 7.1 and compatible with .NET Framework 1.1 / 2.0 or higher.

Requirements

Visual C#, Visual Basic .NET

- Required reference: bin\Suprema.UFExtractor.dll
- Required dll: bin\Suprema.UFExtractor.dll, bin\UFExtractor.dll

Suprema

Suprema namespace

Enumerations

UFE_STATUS	Every function in UFExtractor module returns UFE_STATUS enumeration value
UFE_MODE	UFExtractor.Mode property gives UFS_MODE enumeration value

Classes

UFExtractor	UFExtractor class provides functionality for extracting templates from input images, etc
-----------------------------	--

UFE_STATUS enumeration

Every function in UFExtractor module returns UFE_STATUS enumeration value.

<code>public enum UFE_STATUS : int</code>

Members

UFE_STATUS member name	Code	Meaning
OK	0	Success
ERROR	-1	General error
ERR_NO_LICENSE	-101	System has no license
ERR_LICENSE_NOT_MATCH	-102	License is not match
ERR_LICENSE_EXPIRED	-103	License is expired
ERR_NOT_SUPPORTED	-111	This function is not supported
ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
ERR_NOT_GOOD_IMAGE	-301	Input image is not good
ERR_EXTRACTION_FAILED	-302	Extraction is failed
ERR_UNDEFINED_MODE	-311	Undefined mode
ERR_CORE_NOT_DETECTED	-351	Core is not detected
ERR_CORE_TO_LEFT	-352	Move finger to left
ERR_CORE_TO_LEFT_TOP	-353	Move finger to left-top
ERR_CORE_TO_TOP	-354	Move finger to top
ERR_CORE_TO_RIGHT_TOP	-355	Move finger to right-top
ERR_CORE_TO_RIGHT	-356	Move finger to right
ERR_CORE_TO_RIGHT_BOTTOM	-357	Move finger to right-bottom
ERR_CORE_TO_BOTTOM	-358	Move finger to bottom
ERR_CORE_TO_LEFT_BOTTOM	-359	Move finger to left-bottom

UFE_MODE enumeration

[UFExtractor.Mode](#) property gives UFS_MODE enumeration value.

```
public enum UFE_MODE : int
```

Members

UFE_MODE member name	Code	Meaning
SFR200	1001	Suprema SFR200
SFR300	1002	Suprema SFR300-S
SFR300v2	1003	Suprema SFR300-S(Ver.2)
GENERAL	1004	General scanner type

UFExtractor class

Properties

Mode	Gets or sets mode for configuring extractor
DetectCore	Gets or sets whether detecting core when extracting templates
TemplateSize	Gets or sets template size limitation of the scanner
UseSIF	Gets or sets whether using SIF for templates

Methods

Extract	Extracts a template from input image
SetEncryptionKey	Sets encryption key
EncryptTemplate	Encrypts template
DecryptTemplate	Decrypts template
LoadImageFromBMPFile	Loads image data from bitmap file
LoadImageFromTIFFFile	Loads image data from tiff file
LoadImageFromJPGFile	Loads image data from jpeg file
LoadImageFromBMPPBuffer	Loads image data from bitmap class instance
LoadImageFromWSQFile	Loads image data from wsq file
LoadImageFromWSQBuffer	Loads image data from wsq data buffer
GetImageBufferTo19794_4ImageBuffer	Gets ISO 19794-4 image buffer from raw image data
GetImageBufferToBMPIImageBuffer	Gets BMP image buffer from raw image data
GetImageBufferToJPEGImageBuffer	Gets JPEG image buffer from raw image data
GetImageBufferToJP2ImageBuffer	Gets JP2 image buffer from raw image data
GetErrorString	Gets the error string

Mode property

Gets or sets mode for configuring extractor. Default value is [UFE_MODE.SFR300v2](#).

```
public UFE\_MODE Mode { get; set; }
```

DetectCore property

Gets or sets whether detecting core when extracting templates. Default value is false.

```
public bool DetectCore { get; set; }
```

TemplateSize property

Gets or sets template size limitation of the scanner. The unit is bytes and the value ranges from 256 to 1024 and step size is 32. Default value is 384.

```
public int TemplateSize { get; set; }
```

UseSIF property

Gets or sets whether using SIF (biometric data standard interchange format) for templates.
Default value is false.

```
public bool UseSIF { get; set; }
```

Extract method

Extracts a template from input image.

```
public UFE_STATUS Extract(
    byte[] ImageData,
    int Width,
    int Height,
    int Resolution,
    byte[] Template,
    out int TemplateSize,
    out int EnrollQuality
);
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_LICENSE_NOT_MATCH](#),
[UFE_STATUS.ERR_LICENSE_EXPIRED](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#),
[UFE_STATUS.ERR_NOT_GOOD_IMAGE](#),
[UFE_STATUS.ERR_EXTRACTION_FAILED](#),
[UFE_STATUS.ERR_CORE_NOT_DETECTED](#), [UFE_STATUS.ERR_CORE_TO_LEFT](#),
[UFE_STATUS.ERR_CORE_TO_LEFT_TOP](#), [UFE_STATUS.ERR_CORE_TO_TOP](#),
[UFE_STATUS.ERR_CORE_TO_RIGHT_TOP](#), [UFE_STATUS.ERR_CORE_TO_RIGHT](#),
[UFE_STATUS.ERR_CORE_TO_RIGHT_BOTTOM](#),
[UFE_STATUS.ERR_CORE_TO_BOTTOM](#),
[UFE_STATUS.ERR_CORE_TO_LEFT_BOTTOM](#)

Parameters

ImageData	[in] Byte array holding input image; input image is assumed to be top-down order and 8 bit gray for pixel format
Width	[in] Width of input image; Width must be less than 640
Height	[in] Height of input image; Height must be less than 640
Resolution	[in] Resolution of input image
Template	[out] Receives the template array; The array must be allocated in advance
TemplateSize	[out] Receives the size (in bytes) of Template
EnrollQuality	[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.

SetEncryptionKey method

Sets encryption key.

```
public UFE_STATUS SetEncryptionKey(  
    byte[] Key  
>);
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Key	[in] 32 bytes key array; default key is first byte is 1 and second to 32th byte are all 0
-----	---

EncryptTemplate method

Encrypts template.

```
public UFE_STATUS EncryptTemplate(  
    byte[] TemplateInput,  
    int TemplateInputSize,  
    byte[] TemplateOutput,  
    ref int TemplateOutputSize  
>;
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

TemplateInput	[in] Input template array
TemplateInputSize	[in] Input template size
TemplateOutput	[out] Output template array
TemplateOutputSize	[in / out] Inputs allocated size of output template array; Receives output template size

DecryptTemplate method

Decrypts template.

```
public UFE_STATUS DecryptTemplate(  
    byte[] TemplateInput,  
    int TemplateInputSize,  
    byte[] TemplateOutput,  
    ref int TemplateOutputSize  
>;
```

Possible return values

[UFE STATUS.OK](#), [UFE STATUS.ERROR](#),
[UFE STATUS.ERR INVALID PARAMETERS](#)

Parameters

TemplateInput	[in] Input template array
TemplateInputSize	[in] Input template size
TemplateOutput	[out] Output template array
TemplateOutputSize	[in / out] Inputs allocated size of output template array; Receives output template size

LoadImageFromBMPFile method

Loads image data from bitmap file.

```
public UFE_STATUS LoadImageFromBMPFile(  
    string BMPFileName,  
    out byte[] ImageData,  
    out int Width,  
    out int Height  
>;
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

BMPFileName	[in] Specifies file name to load image data
ImageData	[out] Byte array holding image data; ImageData is top-down order and has 8 bit gray for pixel format
Width	[out] Width of input image
Height	[out] Height of input image

LoadImageFromTIFFfile method

Loads image data from tiff file.

```
public UFE_STATUS LoadImageFromTIFFfile(  
    string TIFFfileName,  
    out byte[] ImageData,  
    out int Width,  
    out int Height  
>;
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

TIFFfileName	[in] Specifies file name to load image data
ImageData	[out] Byte array holding image data; ImageData is top-down order and has 8 bit gray for pixel format
Width	[out] Width of input image
Height	[out] Height of input image

LoadImageFromJPGFile method

Loads image data from jpeg file.

```
public UFE_STATUS LoadImageFromJPGFile(  
    string JPGFileName,  
    out byte[] ImageData,  
    out int Width,  
    out int Height  
>;
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

JPGFileName	[in] Specifies file name to load image data
ImageData	[out] Byte array holding image data; ImageData is top-down order and has 8 bit gray for pixel format
Width	[out] Width of input image
Height	[out] Height of input image

LoadImageFromBMPBuffer method

Loads image data from bitmap class instance.

```
public UFE_STATUS LoadImageFromBMPBuffer(  
    Bitmap bitmap,  
    out byte[] ImageData,  
    out int Width,  
    out int Height  
)
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

bitmap	[in] Specifies a bitmap class instance holding image data
ImageData	[out] Byte array holding image data; ImageData is top-down order and has 8 bit gray for pixel format
Width	[out] Width of input image
Height	[out] Height of input image

LoadImageFromWSQFile method

Loads image data from wsq file.

```
public UFE_STATUS LoadImageFromWSQFile(  
    string WSQFileName,  
    out byte[] ImageData,  
    out int Width,  
    out int Height  
>;
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

WSQFileName	[in] Specifies file name to load image data
ImageData	[out] Byte array holding image data; ImageData is top-down order and has 8 bit gray for pixel format
Width	[out] Width of input image
Height	[out] Height of input image

LoadImageFromWSQBuffer method

Loads image data from wsq byte array.

```
public UFE_STATUS LoadImageFromWSQBuffer(
    byte[] pWSQBuffer,
    int nWSQBufferSize,
    out byte[] ImageData,
    out int Width,
    out int Height
);
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

pWSQBuffer	[in] Byte array holding wsq image data
nWSQBufferSize	[in] Specifies the size of the input wsq buffer
ImageData	[out] Byte array holding image data; ImageData is top-down order and has 8 bit gray for pixel format
Width	[out] Width of input image
Height	[out] Height of input image

GetErrorString method

Gets the error string for specified [UFE_STATUS](#) value.

```
public static UFE\_STATUS GetErrorString(  
    UFE\_STATUS res,  
    out string ErrorString  
);
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

res	[in] Status return value
ErrorString	[out] Receives error string

GetImageBufferTo19794_4ImageBuffer

Make an ISO 19794-4 image from raw image data.

```
UFE_STATUS UFE_API UFE_GetImageBufferTo19794_4ImageBuffer(
    byte[] pImageDataIn,
    int Width,
    int Height,
    out byte[] pImageDataOut,
    out int pImageDataLength
);
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

pImageDataIn	[in] Pointer to raw image data
Width	[in] Width of the raw image data
Height	[in] Height of the raw image data
pImageDataOut	[out] Pointer to output image data; ISO 19794-4 format
pImageDataLength	[out] length of output image data

Examples

Visual C#
<pre>UFE_STATUS ufe_res; byte[] ImageData = null; int Width = 0; int Height = 0; // Load image ufe_res = UFExtractor.LoadImageFromBMPFile("Input File Name", out ImageData, out Width, out Height); if(ufe_res == UFE_STATUS.OK) { } else { // Load image failed return;</pre>

```
}

UFExtractor Extractor = new UFExtractor();

byte[] outputImage;
int outputSize;

Extractor.GetImageBufferTo19794_4ImageBuffer(ImageData, Width, Height, out
outputImage, out outputSize);

// If want to save the image as a file, use System function, which saves the binary data.
using (FileStream fs = File.Create("Output File Name"))
{
    fs.Write(outputImage, 0, outputSize);
    fs.Close();
}
```

GetImageBufferToBMPIImageBuffer

Make a BMP image from raw image data.

```
UFE\_STATUS UFE_API UFE_GetImageBufferToBMPIImageBuffer(
    byte[] pImageDataIn,
    int Width,
    int Height,
    out byte[] pImageDataOut,
    out int pImageDataLength
);
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

pImageDataIn	[in] Pointer to raw image data
Width	[in] Width of the raw image data
Height	[in] Height of the raw image data
pImageDataOut	[out] Pointer to output image data; BMP format
pImageDataLength	[out] length of output image data

Examples

```
Visual C#
UFE_STATUS ufe_res;
byte[] ImageData = null;
int Width = 0;
int Height = 0;

// Load image
ufe_res = UFExtractor.LoadImageFromBMPFile("Input File Name", out ImageData, out
Width, out Height);
if(ufe_res == UFE\_STATUS.OK)
{
}

else
{
    // Load image failed
```

```
    return;
}

UFExtractor Extractor = new UFExtractor();

byte[] outputImage;
int outputSize;

Extractor.GetImageBufferToBMPIImageBuffer(ImageData, Width, Height, out
outputImage, out outputSize);

// If want to save the image as a file, use System function, which saves the binary data.
using (FileStream fs = File.Create("Output File Name"))
{
    fs.Write(outputImage, 0, outputSize);
    fs.Close();
}
```

GetImageBufferToJPEGImageBuffer

Make a JPEG image from raw image data.

```
UFE\_STATUS UFE_API UFE_GetImageBufferToJPEGImageBuffer(
    byte[] pImageDataIn,
    int Width,
    int Height,
    out byte[] pImageDataOut,
    out int pImageDataLength
);
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

pImageDataIn	[in] Pointer to raw image data
Width	[in] Width of the raw image data
Height	[in] Height of the raw image data
pImageDataOut	[out] Pointer to output image data; JPEG format
pImageDataLength	[out] length of output image data

Examples

Visual C#

```
UFE_STATUS ufe_res;
byte[] ImageData = null;
int Width = 0;
int Height = 0;

// Load image
ufe_res = UFExtractor.LoadImageFromBMPFile("Input File Name", out ImageData, out
Width, out Height);
if(ufe_res == UFE\_STATUS.OK)
{
}

else
{
    // Load image failed
```

```
    return;
}

UFExtractor Extractor = new UFExtractor();

byte[] outputImage;
int outputSize;

Extractor.GetImageBufferToJPEGImageBuffer(ImageData, Width, Height, out
outputImage, out outputSize);

// If want to save the image as a file, use System function, which saves the binary data.
using (FileStream fs = File.Create("Output File Name"))
{
    fs.Write(outputImage, 0, outputSize);
    fs.Close();
}
```

GetImageBufferToJP2ImageBuffer

Make a JP2 image from raw image data.

```
UFE\_STATUS UFE_API UFE_GetImageBufferToJP2ImageBuffer(
    byte[] pImageDataIn,
    int Width,
    int Height,
    out byte[] pImageDataOut,
    out int pImageDataLength
);
```

Possible return values

[UFE_STATUS.OK](#), [UFE_STATUS.ERROR](#),
[UFE_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

pImageDataIn	[in] Pointer to raw image data
Width	[in] Width of the raw image data
Height	[in] Height of the raw image data
pImageDataOut	[out] Pointer to output image data; JPEG2000 format
pImageDataLength	[out] length of output image data

Examples

Visual C#

```
UFE_STATUS ufe_res;
byte[] ImageData = null;
int Width = 0;
int Height = 0;

// Load image
ufe_res = UFExtractor.LoadImageFromBMPFile("Input File Name", out ImageData, out
Width, out Height);
if(ufe_res == UFE\_STATUS.OK)
{
}

else
{
    // Load image failed
```

```
    return;
}

UFExtractor Extractor = new UFExtractor();

byte[] outputImage;
int outputSize;

Extractor.GetImageBufferToJP2ImageBuffer(ImageData, Width, Height, out
outputImage, out outputSize);

// If want to save the image as a file, use System function, which saves the binary data.
using (FileStream fs = File.Create("Output File Name"))
{
    fs.Write(outputImage, 0, outputSize);
    fs.Close();
}
```

Suprema.UFDatabase module

Suprema.UFDatabase module provides functionality for managing database, adding / updating / removing / getting templates with user data, etc. Actually, Suprema.UFDatabase module is C# wrapper dll of UFDatabase.dll. The module is created using Visual C# 7.1 and compatible with .NET Framework 1.1 / 2.0 or higher.

Requirements

Visual C#, Visual Basic .NET

- Required reference: bin\Suprema.UFDatabase.dll
- Required dll: bin\Suprema.UFDatabase.dll, bin\UFDatabase.dll

Database table structure

Database table structure could be found in [UFDatabase module](#).

Suprema

Suprema namespace

Enumerations

<u>UFD_STATUS</u>	Every function in UFExtractor module returns UFE_STATUS enumeration value
-----------------------------------	---

Classes

<u>UFDatabase</u>	UFDatabase class provides functionality for managing database, adding / updating / removing / getting templates with user data, etc
-----------------------------------	---

UFD_STATUS enumeration

Every function in UFDatabase module returns UFD_STATUS enumeration value.

```
public enum UFD_STATUS : int
```

Members

UFD_STATUS member name	Code	Meaning
OK	0	Success
ERROR	-1	General error
ERR_NO_LICENSE	-101	System has no license
ERR_LICENSE_NOT_MATCH	-102	License is not match
ERR_LICENSE_EXPIRED	-103	License is expired
ERR_NOT_SUPPORTED	-111	This function is not supported
ERR_INVALID_PARAMETERS	-112	Input parameters are invalid
ERR_SAME_FINGER_EXIST	-501	Same finger exists on database

UFDatabase class

Methods

<u>Open</u>	Opens a database
<u>Close</u>	Closes specified database
<u>AddData</u>	Adds data into the specified database
<u>UpdateDataByUserInfo</u>	Updates the database entry having specified user ID and finger index
<u>UpdateDataBySerial</u>	Updates the database entry having specified serial number
<u>RemoveDataByUserID</u>	Removes the database entries having specified user ID
<u>RemoveDataByUserInfo</u>	Removes the database entries having specified user ID and finger index
<u>RemoveDataBySerial</u>	Removes the database entries having specified serial number
<u>RemoveAllData</u>	Removes all database entries
<u>GetDataNumber</u>	Gets the number of database entries
<u>GetDataByIndex</u>	Gets the database entry having specified index
<u>GetDataByUserInfo</u>	Gets the database entry having specified user ID and finger index
<u>GetDataBySerial</u>	Gets the database entry having specified serial number
<u>GetTemplateListWithSerial</u>	Gets all the template list with corresponding serial number list from specified database
<u>GetErrorString</u>	Gets the error string

Open method

Opens a database using specified connection string.

```
public UFD_STATUS Open(
    string Connection,
    string UserID,
    string Password
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Connection	[in] Specifies ADO connection strings; to connect to an Access file using the JET OLE DB Provider, use "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=mdb_file_path;"; if database is password protected, use "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=mdb_file_path;Jet OLEDB:Database Password=mdb_password;"
UserID	[in] Specifies user ID directly passed to ADO open method (can be null)
Password	[in] Specifies password directly passed to ADO open method (can be null)

Close method

Closes specified database.

```
public UFD\_STATUS Close();
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

AddData method

Adds data into the specified database. If there is a database entry of same user ID with finger index with input data, the function returns [UFD_STATUS.ERR_SAME_FINGER_EXIST](#).

```
public UFD_STATUS AddData(
    string UserID,
    int FingerIndex,
    byte[] Template1,
    int Template1Size,
    byte[] Template2,
    int Template2Size,
    string Memo
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#),
[UFD_STATUS.ERR_SAME_FINGER_EXIST](#)

Parameters

UserID	[in] Specifies user ID Information
FingerIndex	[in] Specifies finger index for Identifying finger
Template1	[in] Specifies first fingerprint template data
Template1Size	[in] Specifies the size of template
Template2	[in] Specifies second fingerprint template data
Template2Size	[in] Specifies the size of template
Memo	[in] Specifies additional user data

UpdateDataByUserInfo method

Updates the database entry having specified user ID and finger index.

```
public UFD_STATUS UpdateDataByUserInfo(
    string UserID,
    int FingerIndex,
    byte[] Template1,
    int Template1Size,
    byte[] Template2,
    int Template2Size,
    string Memo
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

UserID	[in] Specifies user ID Information
FingerIndex	[in] Specifies finger index for Identifying finger
Template1	[in] Specifies first fingerprint template data; null indicates no update for template1
Template1Size	[in] Specifies the size of template; 0 indicates no update for template1
Template2	[in] Specifies second fingerprint template data; null indicates no update for template2
Template2Size	[in] Specifies the size of template; 0 indicates no update for template2
Memo	[in] Specifies additional user data; null indicates no update for memo

UpdateDataBySerial method

Updates the database entry having specified serial number.

```
public UFD_STATUS UpdateDataBySerial(
    int Serial,
    byte[] Template1,
    int Template1Size,
    byte[] Template2,
    int Template2Size,
    string Memo
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Serial	[in] Specifies serial number
Template1	[in] Specifies first fingerprint template data; null indicates no update for template1
Template1Size	[in] Specifies the size of template; 0 indicates no update for template1
Template2	[in] Specifies second fingerprint template data; null indicates no update for template2
Template2Size	[in] Specifies the size of template; 0 indicates no update for template2
Memo	[in] Specifies additional user data; null indicates no update for memo

RemoveDataByUserID method

Removes the database entries having specified user ID.

```
public UFD_STATUS RemoveDataByUserID(  
    string UserID  
>);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

UserID	[in] Specifies user ID Information
--------	------------------------------------

RemoveDataByUserInfo method

Removes the database entries having specified user ID and finger index.

```
public UFD_STATUS RemoveDataByUserInfo(  
    string UserID,  
    int FingerIndex  
>;
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

UserID	[in] Specifies user ID Information
FingerIndex	[in] Specifies finger index for Identifying finger

RemoveDataBySerial method

Removes the database entries having specified serial number.

```
public UFD_STATUS RemoveDataBySerial(  
    int Serial  
>;
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Serial	[in] Specifies serial number
--------	------------------------------

RemoveAllData method

Removes all database entries.

```
public UFD\_STATUS RemoveAllData();
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

GetDataNumber method

Gets the number of database entries.

```
public UFD_STATUS GetDataNumber(  
    [out] int DataNumber  
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

DataNumber	[out] Receives the number of database entries
------------	---

GetDataByIndex method

Gets the database entry having specified index.

```
public UFD_STATUS GetDataByIndex(
    int Index,
    out int Serial,
    out string UserID,
    out int FingerIndex,
    byte[] Template1,
    out int Template1Size,
    byte[] Template2,
    out int Template2Size,
    out string Memo
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Index	[in] Specifies the index
Serial	[out] Receives the serial number
UserID	[out] Receives user ID Information
FingerIndex	[out] Receives finger index
Template1	[out] Receives first fingerprint template data
Template1Size	[out] Receives the size of template
Template2	[out] Receives second fingerprint template data
Template2Size	[out] Receives the size of template
Memo	[out] Receives additional user data

GetDataByUserInfo method

Gets the database entry having specified user ID and finger index.

```
public UFD_STATUS GetDataByUserInfo(
    int Index,
    out int Serial,
    out string UserID,
    out int FingerIndex,
    byte[] Template1,
    out int Template1Size,
    byte[] Template2,
    out int Template2Size,
    out string Memo
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

UserID	[in] Specifies user ID Information
FingerIndex	[in] Specifies finger index
Template1	[out] Receives first fingerprint template data
Template1Size	[out] Receives the size of template
Template2	[out] Receives second fingerprint template data
Template2Size	[out] Receives the size of template
Memo	[out] Receives additional user data

GetDataBySerial method

Gets the database entry having specified serial number.

```
public UFD_STATUS GetDataBySerial(
    int Serial,
    out string UserID,
    out int FingerIndex,
    byte[] Template1,
    out int Template1Size,
    byte[] Template2,
    out int Template2Size,
    out string Memo
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

Serial	[in] Specifies the serial number
UserID	[out] Receives user ID Information
FingerIndex	[out] Receives finger index
Template1	[out] Receives first fingerprint template data
Template1Size	[out] Receives the size of template
Template2	[out] Receives second fingerprint template data
Template2Size	[out] Receives the size of template
Memo	[out] Receives additional user data

GetTemplateListWithSerial method

Gets all the template list with corresponding serial number list from specified database.

```
public UFD\_STATUS GetTemplateListWithSerial(  
    out byte[][] TemplateArray,  
    out int[] TemplateSizeArray,  
    out int TemplateNum,  
    out int[] SerialArray  
);
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

TemplateArray	[out] Receives the template list
TemplateSizeArray	[out] Receives the tempalte size list
TemplateNum	[out] Receives the size of template list
SerialArray	[out] Receives the serial number list

GetErrorString method

Gets the error string for specified [UFD_STATUS](#) value.

```
public static UFD_STATUS GetErrorString(  
    UFD_STATUS res,  
    out string ErrorString  
>;
```

Possible return values

[UFD_STATUS.OK](#), [UFD_STATUS.ERROR](#),
[UFD_STATUS.ERR_INVALID_PARAMETERS](#)

Parameters

res	[in] Status return value
ErrorString	[out] Receives error sting

FAQ

System Issue

Q. What is the difference between the verification and the identification?

A. The verification is 1:1 matching method and the identification is 1:N matching. 1:N matching can be realized using verification several times. But we recommend that customers use API's 'identification' function because it has additional routines in order to speed up for the 1:N matching.

Q. If I have only template data but no fingerprint image, can I execute a matching process?

A. The fingerprint identification need template data only.

Q. The image SDK uses only BMP image format. Is the image SDK available for BMP format only? Is it impossible using other image format?

A. We support the WSQ file format in Image SDK from version 3.2. We recommend that our customers do not use other data formats because of data loss concerns from other formats.

Function & Parameter Issue

Q. I am using the default parameters in enroll procedure and UFS_PARAM_SENSITIVITY=7 but, some fingerprints in the enroll procedures gets low quality and in the verification doesn't recognize them, what can I do about this?.

A. If the sensitivity value is high, your scanner captures an fingerprint image even if the fingerprint area is very small. And this bed quality fingerprint image might cause low quality template. Thus we recommend low sensitivity value for high quality template and getting the fingerprint image on center of the prism.

Q. What is the UFM_RotateTemplate function?

A. The UFM_RotateTemplate rotates a template 180 degree. UFMatcher can match the templates just until 90 degree. If a template is reversed, you should use this function.

Q. What is the UFS_PARAM_DETECT_CORE parameter?

A. If you set 'detect core' option to true value, your scanner will capture an image only when 'fingerprint core' is existed in the image. But it is better that not using this option. Because the fingerprint core often doesn't exist dependently on fingerprint.

Q. What is the UFM_IdentifyMT function?

A. The UFM_IdentifyMT uses the multi-thread method. If you use this function, the matching speed will be increased twice compared with normal UFM_Identify function. But this function should be used in multi-core CPU only.

Q. What is the purpose of the encryption function like UFS_EncryptTemplate function?
A. To restrain exposure of the user data when you accesses the database.

Template Issue

Q. There are three types of template: SUPREMA, ISO and ANSI type. What's the difference between these templates?

A. The matching performance is different. The SUPREMA type shows better performance compared with others. And the ISO and ANSI type show same performance. But the SUPREMA type is our own template format and it is not standard template format.

Q. I send a template which is half empty. The template size is 384 bytes. I found out half of the template data value 0. Why half of value is 0. Why those values are 0?

A. You set the maximum template length as 384 bytes. And the length of template could be changed depending on the condition of fingerprint image. In this case, if the template length is less than 384 bytes, the rest of the empty values of 384 bytes are filled with zeros("zero-padded").

Q. If I use the 'capture' function and the 'extract' function two times for one finger, Will I get exactly the same two fingerprint templates?

A. The 'extract' function generates only one template from one fingerprint image at one time. And even if you capture same finger several times, you will get a different template from the fingerprint images. Because the fingerprint template will be changed depending on the condition and the position of the fingerprint image captured from a finger and the condition and the position of the fingerprint image will be different everytime you scan a finger.

Scanner Issue

Q. If my scanner can not read fingerprint image well, what can I do for that?

A. Adjust the sensitivity value. If the image is not good, adjust the brightness value.

Q. If-only small part of the finger or the other part of the fingerprint like fingernail contacts with the prism, the scanner doesn't work. Is it possible to get a fingerprint image regardless of the image quality or the fingerprint area?

A. Our scanner was not considered other usage except scanning fingerprint. But you can get the image even if there are no fingerprint in the image frame in case. You use the UFS_StartCapturing function. Then, you can get the image frame from the callback function.

Q. How many scanners can be used in one PC?

A. Recommended maximum number of scanners for stable process is four.

Q. I want to know the default setting of the image, such as the image size or the proportions (landscape and portrait).

A. The Basic image setting is specialized depending on the scanner type. In case of BioMini, the size of image is 288 x 320 and the SFR300 and SFR200's image size is 280 x 320.

DataBase Issue

Q. If I use two templates and those two templates contains same part of the finger, can I generate a better fingerprint template by using those two templates at once?

A. one template can be used at one time. But using more than one template will increase the matching success rate. The recognition success rate and the accuracy of the matching performance are increased as enrolling several templates of about one finger. But the matching processing time is getting slower as adding the number of template. The database demo sample is just designed to save 'different' two templates from about one fingerprint. And using the second template of the finger is an option.

Q. How can I use two templates from one finger?

A. 'extract' two times and then add the template1 and the template2 by using UFD_AddData function.

Q. What data do I must save into the database?

A. Only User ID data and template data are needed. The rest of the user data is not an essential.

Q. Does the UFDatabase module uses the mdb only? Can't the UFDatabase module use other databases?

A. The UFDatabase module is a sample to show you that a method about saving and loading the user data and the template into the mdb(Access Database) file. User create own database handling system by themselves.

Q. Why the fingerprint images is not saved in the database.

A. Generally, the fingerprint recognition system doesn't save the fingerprint images. Because the fingerprint image is directly related to the security issue.

Q. What is the finger index field in database sample demo?

A. The finger index field shows a number which is assigned from the left hand little finger to the right hand little finger. This field allows to insert a finger data separately in the same UserID entry.

Performance Issue

Q. In the fingerprint enrollment I am using the default parameters, and I get good enroll quality in 95% of the fingerprints, but the rest 5% is very bad, in some cases I can not get better fingerprint quality, in the case I use two templates can I get better enroll quality?

A. Yes, Using two templates increases the matching performance. And the 'enroll quality' is an important element to get good matching result. Sometimes, matching error could be occurred even if you adjust the security level. The reason of this problem is that the fingerprint templates were enrolled despite the quality of template is too bad. In other words, the enrollment process is very important in fingerprint identification system. Please be reminded that the high quality of templates should be enrolled to the database for avoiding matching fail. Higher than 90 quality is recommended before version 3.3 and higher than 60 quality is recommended from version 3.3.

Q. How can I enhance the brightness of the fingerprint image?

A. If the fingerprint image is not clear, please adjust the brightness value.

Q. How can I increase the matching speed?

A. Use the 'fast mode' option. If your PC supports the dual core, use the UFM_IdentifyMT(multi-thread) function is recommended. And then the matching speed will be twice as fast as before. But If your PC doesn't support the dual core, using the UFM_IdentifyMT function will not affect to the speed.

Q. What is the UFM_PARAM_SECURITY_LEVEL? And dose it influence to the matching performance?

A. The security level is related to the accuracy of the matching result. The high security level decreases the false acceptance(FA) error rate. But the high security level increases the false reject(FR) error rate at the same time. So please adjust the security level based on the scale of your fingerprint database.

Appendix A. Contacts

Headquarters

Suprema Inc.

Address: 16F Parkview office Tower, Jeongja-dong, Bundang-gu, Seongnam, Gyeonggi,
463-863, Korea
Tel: +82-31-783-4502
Fax: +82-31-783-4503

E-mail support

If you wish to directly query a problem in the Suprema PC SDK, send a mail to
support@suprema.com or sales@suprema.com.

Appendix B. Distribution Content

Suprema PC SDK 3.4.1 distribution contains the following directories:

[bin](#)
[docs](#)
[include](#)
[install](#)
[lib](#)
[samples](#)

bin

File Name	Description	Supported Products ¹⁾
Finger1_1.bmp	Sample fingerprint image	PI
Finger1_2.bmp	Sample fingerprint image	PI
Finger2_1.bmp	Sample fingerprint image	PI
Finger2_2.bmp	Sample fingerprint image	PI
Suprema.tlb	Type library for Visual Basic 6.0	PE, PS, PM, PI
Suprema.UFDatabase.dll	Database module wrapper for .NET	PS
Suprema.UFExtractor.dll	Extractor module wrapper for .NET	PI
Suprema.UFMatcher.dll	Matcher module wrapper for .NET	PS, PM, PI
Suprema.UFScanner.dll	Scanner module wrapper for .NET	PE, PS
UFDatabase.dll	Database module	PS
UFDatabase.mdb	Database file	PS
UFE30_DatabaseDemoCS.exe	Database demo for C#	PS
UFE30_DatabaseDemoVBNET.exe	Database demo for VB .NET	PS
UFE30_DatabaseDemoVC60.exe	Database demo for VC++ 6.0	PS
UFE30_DemoCS.exe	Demo for C#	PS
UFE30_DemoVBNET.exe	Demo for VB .NET	PS
UFE30_DemoVC60.exe	Demo for VC++ 6.0	PS
UFE30_EnrollDemoCS.exe	Enroll demo for C#	PE
UFE30_EnrollDemoVBNET.exe	Enroll demo for VB .NET	PE
UFE30_EnrollDemoVC60.exe	Enroll demo for VC++ 6.0	PE
UFE30_ImageDemoCS.exe	Image demo for C#	PI
UFE30_ImageDemoVBNET.exe	Image demo for VB .NET	PI
UFE30_ImageDemoVC60.exe	Image demo for VC++ 6.0	PI
UFE30_MultiScannerDemoVC60.exe	Multi scanner demo	PS
UFExtractor.dll	Extractor module	PI
UFLicense.dat	License file	PE, PS, PM, PI
UFMatcher.dll	Matcher module	PS, PM, PI
UFScanner.dll	Scanner module	PE, PS
UFScanner_IZZIX.dl	Scanner sub module for SFR200	PE, PS
/com/suprema/ufe33/UFExtractorClass.class	com.suprema.ufe33 package, Extractor class	PI
/com/suprema/ufe33/UFMatcherClass.class	com.suprema.ufe33 package, Matcher class	PS, PM, PI
/com/suprema/ufe33/UFScannerClass.class	com.suprema.ufe33 package, Scanner class	PE, PS
/com/suprema/ufe33/BioMiniJniSDK.class	com.suprema.ufe33 package, BioMini class	PS

Suprema PC SDK 3.4.1

¹⁾ PE = [Suprema BioMini Enroll SDK](#), PS = [Suprema BioMini SDK](#), PM = [Suprema Match SDK](#), PI = [Suprema Image SDK](#)

docs

File Name	Description	Supported Products ¹⁾
Suprema_PC_SDK_3.4[3.4.1.0]_Reference_Manual.chm	Reference Manual (Windows help file format)	PE, PS, PM, PI
Suprema_PC_SDK_3.4[3.4.1.0]_Reference_Manual.pdf	Reference Manual (Adobe PDF format)	PE, PS, PM, PI

¹⁾ PE = [Suprema BioMini Enroll SDK](#), PS = [Suprema BioMini SDK](#), PM = [Suprema Match SDK](#), PI = [Suprema Image SDK](#)

include

File Name	Description	Supported Products ¹⁾
UFDatabase.h	Header file for database module	PS
UFExtractor.h	Header file for extractor module	PI
UFMatcher.h	Header file for matcher module	PS, PM, PI
UFScanner.h	Header file for scanner module	PE, PS

¹⁾ PE = [Suprema BioMini Enroll SDK](#), PS = [Suprema BioMini SDK](#), PM = [Suprema Match SDK](#), PI = [Suprema Image SDK](#)

install

File Name	Description	Supported Products ①)
drivers\SFR200\IZZIX.98_	Driver file for SFR200	PE, PS
drivers\SFR200\IZZIX.cat	Driver file for SFR200	PE, PS
drivers\SFR200\IZZIX.inf	Driver file for SFR200	PE, PS
drivers\SFR200\IZZIX.xp_	Driver file for SFR200	PE, PS
drivers\SFR300-S\sfudusb.inf	Driver file for SFR300-S	PE, PS
drivers\SFR300-S\sfudusb.sys	Driver file for SFR300-S	PE, PS
drivers\SFR300-S(Ver.2)\2000_XP\CyLoad.sys	Driver file for SFR300-S(Ver.2), For Windows 2000, XP	PE, PS
drivers\SFR300-S(Ver.2)\2000_XP\CyUSB.sys	Driver file for SFR300-S(Ver.2), For Windows 2000, XP	PE, PS
drivers\SFR300-S(Ver.2)\2000_XP\SFR300V2.inf	Driver file for SFR300-S(Ver.2), For Windows 2000, XP	PE, PS
drivers\SFR300-S(Ver.2)\2000_XP\SFR300V2.spt	Driver file for SFR300-S(Ver.2), For Windows 2000, XP	PE, PS
drivers\SFR300-S(Ver.2)\98_ME\firmware.sys	Driver file for SFR300-S(Ver.2), For Windows 98, ME	PE, PS
drivers\SFR300-S(Ver.2)\98_ME\Sfu-Usb.inf	Driver file for SFR300-S(Ver.2), For Windows 98, ME	PE, PS
drivers\SFR300-S(Ver.2)\98_ME\Sfu-Usb.sys	Driver file for SFR300-S(Ver.2), For Windows 98, ME	PE, PS
drivers\SFR(BioMini & BioMini Plus)\Sup_Fingerprint_Driver_v2.0.1.exe	Universal driver for SFR400, SFR410 and SFR500	PE, PS

①) PE = [Suprema BioMini Enroll SDK](#), PS = [Suprema BioMini SDK](#), PM = [Suprema Match SDK](#), PI = [Suprema Image SDK](#)

lib

File Name	Description	Supported Products ¹⁾
UFDatabase.lib	Library file for database module	PS
UFExtractor.lib	Library file for extractor module	PI
UFMatcher.lib	Library file for matcher module	PS, PM, PI
UFScanner.lib	Library file for scanner module	PE, PS

¹⁾ PE = [Suprema BioMini Enroll SDK](#), PS = [Suprema BioMini SDK](#), PM = [Suprema Match SDK](#), PI = [Suprema Image SDK](#)

samples

Directory Name	Description	Supported Products 1)
VS60\UFE30_DatabaseDemoVB60	Contains database demo sample project for Visual Basic 6.0	PS
VS60\UFE30_DatabaseDemoVC60	Contains database demo sample project for Visual C++ 6.0	PS
VS60\UFE30_DemoVB60	Contains demo sample project for Visual Basic 6.0	PS
VS60\UFE30_DemoVC60	Contains demo sample project for Visual C++ 6.0	PS
VS60\UFE30_EnrollDemoVB60	Contains enroll demo sample project for Visual Basic 6.0	PE
VS60\UFE30_EnrollDemoVC60	Contains enroll demo sample project for Visual C++ 6.0	PE
VS60\UFE30_ImageDemoVB60	Contains image demo sample project for Visual Basic 6.0	PI
VS60\UFE30_ImageDemoVC60	Contains image demo sample project for Visual C++ 6.0	PI
VS60\UFE30_MultiScannerDemoVC60	Contains multi scanner demo sample project for Visual C++ 6.0	PS
VS80\UFE30_DatabaseDemoCS	Contains database demo sample project for Visual C#	PS
VS80\UFE30_DatabaseDemoVBNET	Contains database demo sample project for Visual Basic .NET	PS
VS80\UFE30_DemoCS	Contains demo sample project for Visual C#	PS
VS80\UFE30_DemoVBNET	Contains demo sample project for Visual Basic .NET	PS
VS80\UFE30_EnrollDemoCS	Contains enroll demo sample project for Visual C#	PE
VS80\UFE30_EnrollDemoVBNET	Contains enroll demo sample project for Visual Basic .NET	PE
VS80\UFE30_ImageDemoCS	Contains image demo sample project for Visual C#	PI
VS80\UFE30_ImageDemoVBNET	Contains image demo sample project for Visual Basic .NET	PI
java\JNA\demoUFEJavaJNA	Contains demo sample project for java JNA	PS
java\JNI\demoUFEJavaJNI	Contains demo sample project for java JNI	PS

¹⁾ PE = [Suprema BioMini Enroll SDK](#), PS = [Suprema BioMini SDK](#), PM = [Suprema Match SDK](#), PI = [Suprema Image SDK](#)

Index

- Enroll Quality, 567
- FAQ, 565
- HUFDATABASE, 413
- HUFExtractor, 362
- HUFMatcher, 321
- HUFScanner, 216
- Suprema namespace, 451, 504, 519, 547
- Suprema.UFD_STATUS enumeration, 548
- Suprema.UFD_STATUS.ERR_INVA
LID_PARAMETERS, 548, 567
- Suprema.UFD_STATUS.ERR_LICE
NSE_EXPIRED, 548
- Suprema.UFD_STATUS.ERR_LICE
NSE_NOT_MATCH, 548
- Suprema.UFD_STATUS.ERR_NO_
LICENSE, 548
- Suprema.UFD_STATUS.ERR_NOT
_SUPPORTED, 548
- Suprema.UFD_STATUS.ERR_SAM
E_FINGER_EXIST, 548
- Suprema.UFD_STATUS.ERROR, 548
- Suprema.UFD_STATUS.OK, 548
- Suprema.UFDatabase module, 546
- Suprema.UFDatabase.AddData
method, 552
- Suprema.UFDatabase.Close method, 551
- Suprema.UFDatabase.GetDataByInde
x method, 560
- Suprema.UFDatabase.GetDataBySeri
al method, 562
- Suprema.UFDatabase.GetDataByUse
rInfo method, 561
- Suprema.UFDatabase.GetDataNumbe
r method, 559
- Suprema.UFDatabase.GetErrorString
method, 564
- Suprema.UFDatabase.GetTemplateLi
stWithSerial method, 563
- Suprema.UFDatabase.Open method, 550
- Suprema.UFDatabase.RemoveAllDat
a method, 558
- Suprema.UFDatabase.RemoveDataB
ySerial method, 557
- Suprema.UFDatabase.RemoveDataB
yUserID method, 555
- Suprema.UFDatabase.RemoveDataB
yUserInfo method, 556
- Suprema.UFDatabase.UpdateDataBy
Serial method, 554
- Suprema.UFDatabase.UpdateDataBy
UserInfo method, 553
- Suprema.UFE_MODE enumeration, 521
- Suprema.UFE_MODE.GENERAL, 521
- Suprema.UFE_MODE.SFR200, 521
- Suprema.UFE_MODE.SFR300, 521
- Suprema.UFE_MODE.SFR300v2, 521
- Suprema.UFE_STATUS enumeration, 520
- Suprema.UFE_STATUS.ERR_COR
E_NOT_DETECTED, 520
- Suprema.UFE_STATUS.ERR_COR
E_TO_BOTTOM, 520
- Suprema.UFE_STATUS.ERR_COR
E_TO_LEFT, 520
- Suprema.UFE_STATUS.ERR_COR
E_TO_LEFT_BOTTOM, 520
- Suprema.UFE_STATUS.ERR_COR
E_TO_LEFT_TOP, 520
- Suprema.UFE_STATUS.ERR_COR
E_TO_RIGHT, 520
- Suprema.UFE_STATUS.ERR_COR
E_TO_RIGHT_BOTTOM, 520
- Suprema.UFE_STATUS.ERR_COR
E_TO_RIGHT_TOP, 520
- Suprema.UFE_STATUS.ERR_COR
E_TO_TOP, 520
- Suprema.UFE_STATUS.ERR_EXTR
ACTION_FAILED, 520
- Suprema.UFE_STATUS.ERR_INVA
LID_PARAMETERS, 520
- Suprema.UFE_STATUS.ERR_LICE
NSE_EXPIRED, 520
- Suprema.UFE_STATUS.ERR_LICE
NSE_NOT_MATCH, 520

Suprema.UFE_STATUS.ERR_NO_LICENSE, 520
 Suprema.UFE_STATUS.ERR_NOT_GOOD_IMAGE, 520
 Suprema.UFE_STATUS.ERR_NOT_SUPPORTED, 520
 Suprema.UFE_STATUS.ERR_UNDEFINED_MODE, 520
 Suprema.UFE_STATUS.ERROR, 520
 Suprema.UFE_STATUS.OK, 520
 Suprema.UFExtractor module, 518
 Suprema.UFExtractor.DecryptTemplate method, 530
 Suprema.UFExtractor.DetectCore property, 524
 Suprema.UFExtractor.EncryptTemplate method, 529
 Suprema.UFExtractor.Extract method, 527
 Suprema.UFExtractor.GetErrorString method, 537
 Suprema.UFExtractor.LoadImageFromBMPBuffer method, 534
 Suprema.UFExtractor.LoadImageFromBMPFile method, 531
 Suprema.UFExtractor.LoadImageFromJPGFile method, 533
 Suprema.UFExtractor.LoadImageFromTIFFFile method, 532
 Suprema.UFExtractor.Mode property, 523
 Suprema.UFExtractor.SetEncryptionKey method, 528
 Suprema.UFExtractor.TemplateSize property, 525
 Suprema.UFExtractor.UseSIF property, 526
 Suprema.UMF_STATUS enumeration, 505
 Suprema.UMF_STATUS.ERR_INV_ALID_PARAMETERS, 505
 Suprema.UMF_STATUS.ERR_LICENSE_EXPIRED, 505
 Suprema.UMF_STATUS.ERR_LICENSE_NOT_MATCH, 505
 Suprema.UMF_STATUS.ERR_MACH_ABORTED, 505
 Suprema.UMF_STATUS.ERR_MACH_TIMEOUT, 505
 Suprema.UMF_STATUS.ERR_NO_LICENSE, 505
 Suprema.UMF_STATUS.ERR_NOT_SUPPORTED, 505
 Suprema.UMF_STATUS.ERR_TEMPLATE_TYPE, 505
 Suprema.UMF_STATUS.ERROR, 505
 Suprema.UMF_STATUS.OK, 505
 Suprema.UMFMatcher module, 503
 Suprema.UMFMatcher.AbortIdentify method, 513
 Suprema.UMFMatcher.FastMode property, 507
 Suprema.UMFMatcher.GetErrorString method, 517
 Suprema.UMFMatcher.Identify method, 511
 Suprema.UMFMatcher.IdentifyInit method, 514
 Suprema.UMFMatcher.IdentifyMT method, 511
 Suprema.UMFMatcher.IdentifyNext method, 515
 Suprema.UMFMatcher.RotateTemplate method, 516
 Suprema.UMFMatcher.SecurityLevel property, 508
 Suprema.UMFMatcher.UseSIF property, 509
 Suprema.UMFMatcher.Verify method, 510
 Suprema.UFS_CAPTURE_PROC delegate, 455
 Suprema.UFS_SCANNER_PROC delegate, 454
 Suprema.UFS_SCANNER_TYPE enumeration, 453
 Suprema.UFS_SCANNER_TYPE.SFR200, 453
 Suprema.UFS_SCANNER_TYPE.SFR300, 453
 Suprema.UFS_SCANNER_TYPE.SFR300v2, 453
 Suprema.UFS_STATUS enumeration, 452

Suprema.UFS_STATUS.ERR_ALRE
ADY_INITIALIZED, 452
Suprema.UFS_STATUS.ERR_CAPT
URE_FAILED, 452
Suprema.UFS_STATUS.ERR_CAPT
URE_RUNNING, 452
Suprema.UFS_STATUS.ERR_CORE
NOT_DETECTED, 452
Suprema.UFS_STATUS.ERR_CORE
TO_BOTTOM, 452
Suprema.UFS_STATUS.ERR_CORE
TO_LEFT, 452
Suprema.UFS_STATUS.ERR_CORE
TO_LEFT_BOTTOM, 452
Suprema.UFS_STATUS.ERR_CORE
TO_LEFT_TOP, 452
Suprema.UFS_STATUS.ERR_CORE
TO_RIGHT, 452
Suprema.UFS_STATUS.ERR_CORE
TO_RIGHT_BOTTOM, 452
Suprema.UFS_STATUS.ERR_CORE
TO_RIGHT_TOP, 452
Suprema.UFS_STATUS.ERR_CORE
TO_TOP, 452
Suprema.UFS_STATUS.ERR_DEVI
CE_NUMBER_EXCEED, 452
Suprema.UFS_STATUS.ERR_EXTR
ACTION_FAILED, 452
Suprema.UFS_STATUS.ERR_INVA
LID_PARAMETERS, 452
Suprema.UFS_STATUS.ERR_LICE
NSE_EXPIRED, 452
Suprema.UFS_STATUS.ERR_LICE
NSE_NOT_MATCH, 452
Suprema.UFS_STATUS.ERR_LOA
D_SCANNER_LIBRARY, 452
Suprema.UFS_STATUS.ERR_NO_L
ICENSE, 452
Suprema.UFS_STATUS.ERR_NOT_
GOOD_IMAGE, 452
Suprema.UFS_STATUS.ERR_NOT_
INITIALIZED, 452
Suprema.UFS_STATUS.ERR_NOT_
SUPPORTED, 452
Suprema.UFS_STATUS.ERROR,
452
Suprema.UFS_STATUS.OK, 452
Suprema.UFScanner module, 450
Suprema.UFScanner.AbortCapturing
method, 488
Suprema.UFScanner.Brightness
property, 473
Suprema.UFScanner.CaptureEvent
event, 470
Suprema.UFScanner.CaptureSingleI
mage method, 486
Suprema.UFScanner.ClearCaptureIm
ageBuffer method, 499
Suprema.UFScanner.DecryptTemplat
e method, 492
Suprema.UFScanner.DetectCore
property, 477
Suprema.UFScanner.DrawCaptureIm
ageBuffer method, 494
Suprema.UFScanner.EncryptTemplat
e method, 491
Suprema.UFScanner.Extract method,
489
Suprema.UFScanner.GetCaptureImag
eBuffer method, 493
Suprema.UFScanner.GetErrorString
method, 500
Suprema.UFScanner.Handle property,
484
Suprema.UFScanner.ID property, 471
Suprema.UFScanner.IsCapturing
property, 483
Suprema.UFScanner.IsFingerOn
property, 482
Suprema.UFScanner.IsSensorOn
property, 481
Suprema.UFScanner.SaveCaptureIma
geBufferToBMP method, 496
Suprema.UFScanner.SaveCaptureIma
geBufferToJPG method, 498
Suprema.UFScanner.SaveCaptureIma
geBufferToTIF method, 497
Suprema.UFScanner.ScannerType
property, 480
Suprema.UFScanner.Sensitivity
property, 474
Suprema.UFScanner.Serial property,
476
Suprema.UFScanner.SetEncryptionK
ey method, 490
Suprema.UFScanner.SetScanner
method, 485

Suprema.UFScanner.StartCapturing
 method, 487

Suprema.UFScanner.TemplateSize
 property, 478

Suprema.UFScanner.Timeout
 property, 472

Suprema.UFScanner.UseSIF property,
 479

Suprema.UFScannerCaptureEventArgs class, 457

Suprema.UFScannerManager.Init
 method, 462

Suprema.UFScannerManager.ScannerEvent event, 461

Suprema.UFScannerManager.ScannerList.Count property, 466

Suprema.UFScannerManager.ScannerList.Item property, 467

Suprema.UFScannerManager.Scanners property, 460

Suprema.UFScannerManager.UFScannerManager constructor, 459

Suprema.UFScannerManager.Uninit
 method, 464

Suprema.UFScannerManager.Update
 method, 463

Suprema.UFScannerManagerScannerEventArgs class, 456

UFD_AddData, 418

UFD_Close, 416

UFD_ERR_INVALID_PARAMETERS, 413

UFD_ERR_LICENSE_EXPIRED, 413

UFD_ERR_LICENSE_NOT_MATCH, 413

UFD_ERR_NO_LICENSE, 413

UFD_ERR_NOT_SUPPORTED, 413

UFD_ERR_SAME_FINGER_EXIST, 413

UFD_ERROR, 413

UFD_GetDataByIndex, 435

UFD_GetDataBySerial, 440

UFD_GetDataByUserInfo, 438

UFD_GetDataNumber, 433

UFD_GetErrorString, 448

UFD_GetTemplateListWithSerial,
 445

UFD_GetTemplateNumber, 443

UFD_OK, 413

UFD_Open, 414

UFD_RemoveAllData, 431

UFD_RemoveDataBySerial, 429

UFD_RemoveDataByUserID, 425

UFD_RemoveDataByUserInfo, 427

UFD_STATUS, 413

UFD_UpdateDataBySerial, 423

UFD_UpdateDataByUserInfo, 420

UFDatabase module, 412

UFE_Create, 363

UFE_DecryptTemplate, 387

UFE_Delete, 365

UFE_EncryptTemplate, 384

UFE_ERR_CORE_NOT_DETECTE
 D, 361

UFE_ERR_CORE_TO_BOTTOM,
 361

UFE_ERR_CORE_TO_LEFT, 361

UFE_ERR_CORE_TO_LEFT_BOT
 TOM, 361

UFE_ERR_CORE_TO_LEFT_TOP,
 361

UFE_ERR_CORE_TO_RIGHT, 361

UFE_ERR_CORE_TO_RIGHT_BO
 TTOM, 361

UFE_ERR_CORE_TO_RIGHT_TOP,
 361

UFE_ERR_CORE_TO_TOP, 361

UFE_ERR_EXTRACTION_FAILED,
 361

UFE_ERR_INVALID_PARAMETERS, 361

UFE_ERR_LICENSE_EXPIRED,
 361

UFE_ERR_LICENSE_NOT_MATCH
 H, 361

UFE_ERR_NO_LICENSE, 361

UFE_ERR_NOT_GOOD_IMAGE,
 361

UFE_ERR_NOT_SUPPORTED, 361

UFE_ERR_UNDEFINED_MODE,
 361

UFE_ERROR, 361

UFE_Extract, 379

UFE_GetErrorString, 400

UFE_GetMode, 367

UFE_GetParameter, 371

UFE_LoadImageFromBMPBuffer,
393
 UFE_LoadImageFromBMPFile, 390
 UFE_MODE_GENERAL, 362
 UFE_MODE_SFR200, 362
 UFE_MODE_SFR300, 362
 UFE_MODE_SFR300v2, 362
 UFE_OK, 361
 UFE_PARAM_DETECT_CORE,
362
 UFE_PARAM_TEMPLATE_SIZE,
362
 UFE_PARAM_USE_SIF, 362
 UFE_SetEncryptionKey, 382
 UFE_SetMode, 369
 UFE_SetParameter, 374
 UFE_STATUS, 361
 UFE30_DatabaseDemo, 153
 UFE30_Demo_Usage, 55
 UFE30_EnrollDemo, 192
 UFE30_ImageDemo, 115
 UFE30_MultiScannerDemo, 198
 UFExtractor module, 360
 UFM_AbortIdentify, 340
 UFM_Create, 322
 UFM_Delete, 324
 UFM_ERR_INVALID_PARAMETERS, 320
 UFM_ERR_LICENSE_EXPIRED,
320
 UFM_ERR_LICENSE_NOT_MATCH, 320
 UFM_ERR_MATCH_ABORTED,
320
 UFM_ERR_MATCH_TIMEOUT,
320
 UFM_ERR_NO_LICENSE, 320
 UFM_ERR_NOT_SUPPORTED,
320
 UFM_ERR_TEMPLATE_TYPE,
320
 UFM_ERROR, 320
 UFM_GetErrorString, 354
 UFM_GetParameter, 326
 UFM_Identify, 336
 UFM_IdentifyInit, 343
 UFM_IdentifyMT, 336
 UFM_IdentifyNext, 346
 UFM_OK, 320
 UFM_PARAM_FAST_MODE, 320
 UFM_PARAM_SECURITY_LEVEL,
320
 UFM_PARAM_USE_SIF, 321
 UFM_RotateTemplate, 351
 UFM_SetParameter, 329
 UFM_STATUS, 320
 UFM_Verify, 332
 UFMatcher module, 319
 UFS_AbortCapturing, 268
 UFS_CAPTURE_PROC, 216
 UFS_CaptureSingleImage, 55, 260
 UFS_ClearCaptureImageBuffer, 308
 UFS_DecryptTemplate, 282
 UFS_DrawCaptureImageBuffer, 293
 UFS_EncryptTemplate, 278
 UFS_ERR_ALREADY_INITIALIZE
D, 214
 UFS_ERR_CAPTURE_FAILED,
214
 UFS_ERR_CAPTURE_RUNNING,
214
 UFS_ERR_CORE_NOT_DETECTE
D, 214
 UFS_ERR_CORE_TO_BOTTOM,
214
 UFS_ERR_CORE_TO_LEFT, 214
 UFS_ERR_CORE_TO_LEFT_BOTT
OM, 214
 UFS_ERR_CORE_TO_LEFT_TOP,
214
 UFS_ERR_CORE_TO_RIGHT, 214
 UFS_ERR_CORE_TO_RIGHT_BO
TTOM, 214
 UFS_ERR_CORE_TO_RIGHT_TOP,
214
 UFS_ERR_CORE_TO_TOP, 214
 UFS_ERR_DEVICE_NUMBER_EX
CEED, 214
 UFS_ERR_EXTRACTION_FAILED,
214
 UFS_ERR_INVALID_PARAMETERS, 214
 UFS_ERR_LICENSE_EXPIRED,
214
 UFS_ERR_LICENSE_NOT_MATCH
H, 214
 UFS_ERR_LOAD_SCANNER_LIB
RARY, 214

UFS_ERR_NO_LICENSE, 214
UFS_ERR_NOT_GOOD_IMAGE,
 214
UFS_ERR_NOT_INITIALIZED, 214
UFS_ERR_NOT_SUPPORTED, 214
UFS_ERROR, 214
UFS_Extract, 271
UFS_GetCaptureImageBuffer, 289
UFS_GetCaptureImageBufferInfo,
 286
UFS_GetErrorString, 310
UFS_GetParameter, 246
UFS_GetScannerHandle, 231
UFS_GetScannerHandleByID, 234
UFS_GetScannerID, 240
UFS_GetScannerIndex, 237
UFS_GetScannerNumber, 229
UFS_GetScannerType, 243
UFS_Init, 218
UFS_IsCapturing, 265
UFS_IsFingerOn, 257
UFS_IsSensorOn, 254
UFS_OK, 214
UFS_PARAM_BRIGHTNESS, 215
UFS_PARAM_DETECT_CORE,
 215
UFS_PARAM_DETECT_FAKE,
 215
UFS_PARAM_SENSITIVITY, 215
UFS_PARAM_SERIAL, 215
UFS_PARAM_TEMPLATE_SIZE,
 215
UFS_PARAM_TIMEOUT, 215
UFS_RemoveScannerCallback, 227
UFS_SaveCaptureImageBufferToBM
 P, 299
UFS_SCANNER_PROC, 216
UFS_SCANNER_TYPE_SFR200,
 215
UFS_SCANNER_TYPE_SFR300,
 215
UFS_SCANNER_TYPE_SFR300v2,
 215
UFS_SetEncryptionKey, 275
UFS_SetParameter, 250
UFS_SetScannerCallback, 224
UFS_StartCapturing, 262
UFS_STATUS, 214
UFS_Uninit, 222
UFS_Update, 220
UFScanner module, 213