UniFinger Engine SDK Reference Manual

Version 3.0.0

Copyright (C) 2007 Suprema Inc.

Table of Contents

Table of Contents 1
Chapter 1. Introduction 8
- Modules 8
- Products 8
- Licensing 8
- Supported development tools and languages 8
- Development system requirements 9

Chapter 2. What's new 10
- Version 3.0.0 10

Chapter 3. Tutorial 11
- Enroll fingerprints from scanner 12
  - 0. Required product 12
  - 1. Preliminaries 12
  - 2. Initialize scanner module and check number of scanners 13
  - 3. Get first scanner 14
  - 4. Set parameters 15
  - 5. Capture image and extract template 16
  - 6. Uninitialize scanner module 18
- Verification 20
  - 0. Required product 20
  - 1. Preliminaries 20
  - 2. Create matcher 21
  - 3. Set parameters 22
  - 4. Verify 22
  - 5. Delete matcher 25
- Identification 26
  - 0. Required product 26
  - 1. Preliminaries 26
  - 2. Create matcher 27
  - 3. Set parameters 28
  - 4. Identify 29
  - 5. Delete matcher 32
- Manage database 34
  - 0. Required product 34
  - 1. Preliminaries 34
  - 2. Open database 35
  - 3. Add one database entry 36
  - 4. Get first database entry 38
  - 5. Close database 40
- Enroll fingerprints from image 41
  - 0. Required product 41
  - 1. Preliminaries 41

2. Create extractor 42
3. Set parameters 43
4. Load image and extract template 44
5. Delete matcher 46
Chapter 4. Sample 47
Before running sample applications 47
UFE30_EnrollDemo 48
Required products 48
Available languages 48
UFE30_Demo 49
Required products 49
Available languages 49
UFE30_DatabaseDemo 50
Required products 50
Available languages 50
UFE30_ImageDemo 51
Required products 51
Available languages 51
UFE30_MultiScannerDemo 52
Required products 52
Available languages 52
Reference 53
Chapter 5. Reference 53
UFScanner module 54
Requirements 54
Supported scanners 54
Definitions 55
Status return value (UFS_STATUS) 55
Parameters 56
Scanner type 56
Scanner handle 56
Scanner callback function 56
Capture callback function 57
UFS_Init 59
UFS_Update 60
UFS_Uninit 61
UFS_SetScannerCallback 62
UFS_RemoveScannerCallback 64
UFS_GetScannerNumber 65
UFS_GetScannerHandle 67
UFS_GetScannerHandleByID 69
UFS_GetScannerIndex 71
UFS_GetScannerID 73
UFS_GetScannerType 75
UFS_GetParameter 77
UFS_SetParameter 80
UFS_IsSensorOn 82

UFS_IsFingerOn 84
UFS_CaptureSingleImage 86
UFS_StartCapturing 88
UFS_IsCapturing 90
UFS_AbortCapturing 92
UFS_Extract 94
UFS_SetEncryptionKey 96
UFS_EncryptTemplate 98
UFS_DecryptTemplate 100
UFS_GetCaptureImageBufferInfo 102
UFS_GetCaptureImageBuffer 104
UFS_DrawCaptureImageBuffer 106
UFS_SaveCaptureImageBufferToBMP 108
UFS_ClearCaptureImageBuffer 110
UFS_GetErrorString 112
UFMatcher module 114
Requirements 114
Definitions 115
Status return value (UFM_STATUS) 115
Parameters 115
Matcher handle 116
UFM_Create 117
UFM_Delete 119
UFM_GetParameter 121
UFM_SetParameter 123
UFM_Verify 125
UFM_Identify, UFM_IdentifyMT 128
UFM_AbortIdentify 132
UFM_IdentifyInit 134
UFM_IdentifyNext 136
UFM_RotateTemplate 139
UFM_GetErrorString 141
UFExtractor module 143
Requirements 143
Definitions 144
Status return value (UFE_STATUS) 144
Parameters 144
Mode 145
Extractor handle 145
UFE_Create 146
UFE_Delete 148
UFE_GetMode 150
UFE_SetMode 152
UFE_GetParameter 154
UFE_SetParameter 156
UFE_Extract 158
UFE_SetEncryptionKey 161

UFE_EncryptTemplate 163
UFE_DecryptTemplate 165
UFE_LoadImageFromBMPFile 167
UFE_LoadImageFromBMPBuffer 169
UFE_GetErrorString 171
UFDatabase module 173
Requirements 173
Database table structure 173
Definitions 174
Status return value (UFD_STATUS) 174
Database handle 174
UFD_Open 175
UFD_Close 177
UFD_AddData 179
UFD_UpdateDataByUserInfo 182
UFD_UpdateDataBySerial 185
UFD_RemoveDataByUserID 188
UFD_RemoveDataByUserInfo 190
UFD_RemoveDataBySerial 192
UFD_RemoveAllData 194
UFD_GetDataNumber 196
UFD_GetDataByIndex 198
UFD_GetDataByUserInfo 201
UFD_GetDataBySerial 204
UFD_GetTemplateNumber 207
UFD_GetTemplateListWithSerial 209
UFD_GetErrorString 212
Chapter 6. Reference (.NET) 214
Suprema.UFScanner module 215
Requirements 215
Supported scanners 215
Suprema 216
Suprema namespace 216
UFS_STATUS enumeration 217
UFS_SCANNER_TYPE enumeration 218
UFS_SCANNER_PROC delegate 219
UFS_CAPTURE_PROC delegate 220
UFScannerManagerScannerEventArgs class 221
UFScannerCaptureEventArgs class 222
UFScannerManager class 223
UFScannerManager constructor 224
Scanners property 225
ScannerEvent event 226
Init method 227
Update method 228
Uninit method 229
UFScannerManager.ScannerList class 230

Count property 231
Item property 232
UFScanner class 233
CaptureEvent event 235
ID property 236
Timeout property 237
Brightness property 238
Sensitivity property 239
Serial property 240
DetectCore property 241
TemplateSize property 242
UseSIF property 243
ScannerType property 244
IsSensorOn property 245
IsFingerOn property 246
IsCapturing property 247
Handle property 248
SetScanner method 249
CaptureSingleImage method 250
StartCapturing method 251
AbortCapturing method 252
Extract method 253
SetEncryptionKey method 254
EncryptTemplate method 255
DecryptTemplate method 256
GetCaptureImageBuffer method 257
DrawCaptureImageBuffer method 258
SaveCaptureImageBufferToBMP method 259
SaveCaptureImageBufferToTIF method 260
SaveCaptureImageBufferToJPG method 261
ClearCaptureImageBuffer method 262
GetErrorString method 263
Suprema.UFMatcher module 264
Requirements 264
Suprema 265
Suprema namespace 265
UFM_STATUS enumeration 266
UFMatcher class 267
FastMode property 268
SecurityLevel property 269
UseSIF property 270
Verify method 271
Identify, IdentifyMT method 272
AbortIdentify method 274
IdentifyInit method 275
IdentifyNext method 276
RotateTemplate method 277

GetErrorString method 278
Suprema.UFExtractor module 279
Requirements 279
Suprema 280
Suprema namespace 280
UFE_STATUS enumeration 281
UFE_MODE enumeration 282
UFExtractor class 283
Mode property 284
DetectCore property 285
TemplateSize property 286
UseSIF property 287
Extract method 288
SetEncryptionKey method 290
EncryptTemplate method 291
DecryptTemplate method 292
LoadImageFromBMPFile method 293
LoadImageFromTIFFile method 294
LoadImageFromJPGFile method 295
LoadImageFromBMPBuffer method 296
GetErrorString method 297
Suprema.UFDatabase module 298
Requirements 298
Database table structure 298
Suprema 299
Suprema namespace 299
UFD_STATUS enumeration 300
UFDatabase class 301
Open method 302
Close method 303
AddData method 304
UpdateDataByUserInfo method 305
UpdateDataBySerial method 306
RemoveDataByUserID method 307
RemoveDataByUserInfo method 308
RemoveDataBySerial method 309
RemoveAllData method 310
GetDataNumber method 311
GetDataByIndex method 312
GetDataByUserInfo method 313
GetDataBySerial method 314
GetTemplateListWithSerial method 315
GetErrorString method 316
Appendix A. Contacts 317
Headquarters 317
E-mail support 317
Appendix B. Distribution Content 318

bin 319
docs 320
include 321
install 322
lib 323
samples 324
Index 325

Chapter 1. Introduction

Modules

UniFinger Engine SDK 3.0 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 UniFinger Engine SDK 3.0 is divided into following products,

Product name	Supported modules
UniFinger Engine SFR Enroll SDK	UFScanner
UniFinger Engine SFR SDK	UFScanner, UFMatcher, UFDatabase
UniFinger Engine Match SDK	UFMatcher
UniFinger Engine Image SDK	UFMatcher, UFExractor

Licensing

Every UniFinger Engine SDK module checks a license file when it is initialized. The license file is named as **UFLicense.dat** and it should be located in the same directory with UniFinger Engine SDK modules. 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

Development system requirements

- Pentium-compatible 500MHz processor or better
- Microsoft Windows 98 / ME / 2000 / XP / 2003
- Fingerprint scanner driver for using UFScanner module

Chapter 2. What's new

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

Chapter 3. Tutorial

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

Topics
Enroll fingerprints from scanner
Verification
Identification
Manage database
Enroll fingerprints from image

Enroll fingerprints from scanner

0. Required product

UniFinger Engine SFR Enroll SDK or UniFinger Engine SFR 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
```

2. Initialize scanner module and check number of scanners

Visual C++

```
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
```

Visual Basic 6.0

```
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
```

Visual C#

```
UFS_STATUS ufs_res;
UFScannerManager ScannerManager;
int nScannerNumber;
```

```
// 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
```

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)
```

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;
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;
```

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;
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
```

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

' Unitialize 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()
```

Verification

0. Required product

UniFinger Engine SFR SDK or UniFinger Engine Match SDK or UniFinger Engine 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
```

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
```

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
```

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 Template2ize 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 = UFM_Verify(hMatcher, Template1, Template1Size,
Template2, Template2size, &bVerifySucceed);
ufm_res = Matcher.Verify(Template1, Template1Size,
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
```

5. Delete matcher

Visual C++

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

// Delete matcher
ufm_res = UFM_Create(&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
```

Identification

0. Required product

UniFinger Engine SFR SDK or UniFinger Engine Match SDK or UniFinger Engine 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;
```

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
' Set maximum template number to 50 (number depends on
application)
Const 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
```

3. Set parameters

Visual C++

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

// 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
```

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
```

5. Delete matcher

Visual C++

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

// Delete matcher
ufm_res = UFM_Create(&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
```

Manage database

0. Required product

UniFinger Engine SFR 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

```
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
```

3. Add one database entry

Visual C++

```
// 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);
```

Visual Basic 6.0

```
' 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
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();
```

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

UniFinger Engine 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
```

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
```

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
```

Chapter 4. Sample

UniFinger Engine SDK supplies various type of sample applications as follows,

Samples	Remarks
UFR30_EnrollDemo	Provides the basic usage about managing scanners and executing enrollment
UFR30_Demo	Provides the basic usage about managing scanners and executing enrollment, verification and identification
UFE30_DatabaseDemo	Provides the demo about managing database
UFE30_ImageDemo	Provides the demo about extracting templates from images and matching two templates
UFE30_MultiScannerDemo	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 UniFinger Engine 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 UniFinger Engine module DLLs.

UFE30_EnrollDemo

UFE30_EnrollDemo provides the basic usage about managing scanners and executing enrollment. This program uses UFScanner module.

Required products

Products	Remarks
UniFinger Engine SFR 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_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
UniFinger Engine SFR 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

UFE30_DatabaseDemo

UFE30_DatabaseDemo provides the demo about managing database. This program uses UFDatabase, UFScanner, and UFMatcher modules.

Required products

Products	Remarks
UniFinger Engine SFR 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_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
UniFinger Engine 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_MultiScannerDemo

UFE30_MultiScannerDemo provides the demo about using multiple scanners simultaneously. This program uses UFScanner and UFMatcher modules.

Required products

Products	Remarks
UniFinger Engine SFR SDK	No restriction

Available languages

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

Reference

Chapter 5. Reference

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

- Visual C++ 6.0 / 7.0 / 7.1 / 8.0
- Visual Basic 6.0 [1)]

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 UniFinger Engine SDK 3.0 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 UniFinger Engine 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++
• Required header: include\UFScanner.h • Required lib: lib\UFScanner.lib • Required dll: bin\UFScanner.dll

Visual Basic 6.0
• 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 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

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;
```

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
);
```

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(
        HUFScanner hScanner,
        int bFingerOn,
        unsigned char* pImage,
        int nWidth,
        int nHeight,
        int nResolution,
        void* 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();
```

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
```

UFS_Update

Enforces scanner module to update current connectivity of scanners.

```
UFS_STATUS UFS_API 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
```

UFS_Uninit

Un-initializes scanner module.

```
UFS_STATUS UFS_API 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
```

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
);
```

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
}
```

UFS_RemoveScannerCallback

Removes registered scanner callback function.

```
UFS_STATUS UFS_API 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
}
```

UFS_GetScannerNumber

Gets the number of scanners.

```
UFS_STATUS UFS_API 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
```

UFS_GetScannerHandle

Gets scanner handle using scanner index.

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

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
```

UFS_GetScannerHandleByID

Gets scanner handle using scanner ID.

```
UFS_STATUS UFS_API UFS_GetScannerHandleByID(
        const char* szScannerID,
        HUFScanner* 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++

```
UFS_STATUS ufs_res;
char strID[64];
HUFScanner hScanner;

// Assign scanner ID to strID
// Scanner ID can be retrieved using UFS_GetScannerID function

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
```

UFS_GetScannerIndex

Gets scanner index assigned to scanner handle.

```
UFS_STATUS UFS_API UFS_GetScannerIndex(
        HUFScanner 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;
HUFScaner hScanner;
int nScannerIndex;

// Get hScanner handle

ufs_res = UFS_GetScannerIndex(hScanner, &nScannerIndex);
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
```

UFS_GetScannerID

Gets scanner ID assigned to scanner handle.

```
UFS_STATUS UFS_API UFS_GetScannerID(
        HUFScanner hScanner,
        char* 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++

```
UFS_STATUS ufs_res;
HUFScaner hScanner;
char strID[64]; // Should be larger than 33 bytes

// 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
```

UFS_GetScannerType

Gets scanner type assigned to scanner handle.

```
UFS_STATUS UFS_API UFS_GetScannerType(
        HUFScanner 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;
HUFScaner 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
```

UFS_GetParameter

Gets parameter value.

```
UFS_STATUS UFS_API UFS_GetParameter(
        HUFScanner hScanner,
        int nParam,
        void* 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;
HUFScaner 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
```

UFS_SetParameter

Sets parameter value.

```
UFS_STATUS UFS_API UFS_SetParameter(
        HUFScanner hScanner,
        int nParam,
        void* 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;
HUFScaner 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
```

UFS_IsSensorOn

Checks the scanner is connected or not.

```
UFS_STATUS UFS_API UFS_IsSensorOn(
        HUFScanner 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++

```
UFS_STATUS ufs_res;
HUFScaner hScanner;
int bSensorOn;

// Get hScanner handle

ufs_res = UFS_IsSensorOn(hScanner, &bSensorOn);
if (ufs_res == UFS_OK) {
        // UFS_IsSensorOn is succeeded
} else {
        // 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
```

UFS_IsFingerOn

Checks a finger is placed on the scanner or not.

```
UFS_STATUS UFS_API UFS_IsFingerOn(
        HUFScanner 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++

```
UFS_STATUS ufs_res;
HUFScaner hScanner;
int bFingerOn;

// Get hScanner handle

ufs_res = UFS_IsFingerOn(hScanner, &bFingerOn);
if (ufs_res == UFS_OK) {
        // UFS_IsFingerOn is succeeded
} else {
        // 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
```

UFS_CaptureSingleImage

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

```
UFS_STATUS UFS_API UFS_CaptureSingleImage(
        HUFScanner 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;
HUFScaner 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
```

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
);
```

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 UFS_CAPTURE_PROC function which receives capture events
pParam	[in] Pointer to the capture callback data which will be transmitted with a capture callback event

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;
HUFScaner 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
}
```

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
);
```

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++

```
UFS_STATUS ufs_res;
HUFScaner hScanner;
int bCapturing;

// Get hScanner handle

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
```

UFS_AbortCapturing

Aborts capturing which is started by UFS_CaptureSingleImage or UFS_StartCapturing.

```
UFS_STATUS UFS_API UFS_AbortCapturing(
        HUFScanner 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;
HUFScaner 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
```

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
);
```

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 using UFS_SetParameter function
// Default value is 384 bytes
#define MAX_TEMPLATE_SIZE 384
```

```
UFS_STATUS ufs_res;
HUFScaner 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 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
```

UFS_SetEncryptionKey

Sets encryption key.

```
UFS_STATUS UFS_API UFS_SetEncryptionKey(
        HUFScanner hScanner,
        unsigned char* 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++

```
UFS_STATUS ufs_res;
HUFScaner hScanner;
unsigned char UserKey[32];

// Get hScanner handle

// 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
```

UFS_EncryptTemplate

Encrypts template.

```
UFS_STATUS UFS_API UFS_EncryptTemplate(
        HUFScanner hScanner,
        unsigned char* pTemplateInput,
        int nTemplateInputSize,
        unsigned char* 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;
HUFScaner 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
```

UFS_DecryptTemplate

Decrypts template.

```
UFS_STATUS UFS_API UFS_DecryptTemplate(
        HUFScanner hScanner,
        unsigned char* pTemplateInput,
        int nTemplateInputSize,
        unsigned char* 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;
HUFScaner 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
```

UFS_GetCaptureImageBufferInfo

Gets the information of the capture image buffer.

```
UFS_STATUS UFS_API UFS_GetCaptureImageBufferInfo(
        HUFScanner 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;
HUFScaner 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
```

UFS_GetCaptureImageBuffer

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

```
UFS_STATUS UFS_API UFS_GetCaptureImageBuffer(
        HUFScanner hScanner,
        unsigned char* 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;
HUFScaner 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(pImageBuffer)
```

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
```

UFS_DrawCaptureImageBuffer

Draws the fingerprint image which is acquired using UFS_CaptureSingleImage() or UFS_StartCapturing().

```
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 of not

See also

UFS_CaptureSingleImage, UFS_StartCapturing, UFS_GetCaptureImageBufferInfo, UFS_GetCaptureImageBuffer, UFS_SaveCaptureImageBufferToBMP, UFS_ClearCaptureImageBuffer

Examples

Visual C++

```
UFS_STATUS ufs_res;
HUFScaner 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 As Long

' 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_STATUS.OK) Then
        ' UFS_DrawCaptureImageBuffer is succeeded
Else
    ' UFS_DrawCaptureImageBuffer 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
);
```

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++

```
UFS_STATUS ufs_res;
HUFScaner hScanner;
char szFileName[128];

// Get hScanner handle

// 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
```

UFS_ClearCaptureImageBuffer

Clears the capture image buffer.

```
UFS_STATUS UFS_API UFS_ClearCaptureImageBuffer(
        HUFScanner 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;
HUFScaner 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
```

UFS_GetErrorString

Gets the error string for specified UFS_STAUS value.

```
UFS_STATUS UFS_API UFS_GetErrorString(
        UFS_STATUS res,
        char* 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 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
```

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

Parameters

UFM_GetParameter(), UFM_SetParameter() functions use parameters defined as follows,

<table>
<tr><th>Parameter value definition</th><th>Code</th><th>Meaning</th><th>Default value</th></tr>
<tr><td>UFM_PARAM_FAST_MODE</td><td>301</td><td>Fast Mode (0: not use fast mode, 1: use fast mode)</td><td>1</td></tr>
<tr><td>UFM_PARAM_SECURITY_LEVEL</td><td>302</td><td><table><tr><th>Level</th><th>False Accept Ratio (FAR)</th></tr><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></table></td><td>4</td></tr>
</table>

		<table><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>	
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;
```

UFM_Create

Creates a matcher.

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

UFM_Delete

Deletes specified matcher.

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

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
```

UFM_GetParameter

Gets parameter value.

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

UFM_SetParameter

Sets parameter value.

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

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
);
```

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
```

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++

```
// 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) {
        // 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
Dim i 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
);
```

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
```

UFM_IdentifyInit

Initializes identify with input template.

```
UFM_STATUS UFM_API UFM_IdentifyInit(
        HUFMatcher pMatcher,
        unsigned char* 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
```

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
);
```

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
```

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
);
```

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
```

UFM_GetErrorString

Gets the error string for specified UFM_STAUS value.

```
UFM_STATUS UFM_API UFM_GetErrorString(
        UFM_STATUS res,
        char* 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 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
```

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

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	384

		size)	
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;
```

UFE_Create

Creates an extractor.

```
UFE_STATUS UFE_API UFE_Create(
        HUFExtractor* phExtractor
);
```

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
```

UFE_Delete

Deletes specified extractor.

```
UFE_STATUS UFE_API UFE_Delete(
        HUFExtractor 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
```

UFE_GetMode

Gets mode of the specified extractor.

```
UFE_STATUS UFE_API UFE_GetMode(
        HUFExtractor hExtractor,
        int* 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++

```
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
}
```

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
```

UFE_SetMode

Sets mode of the specified extractor.

```
UFE_STATUS UFE_API UFE_SetMode(
        HUFExtractor hExtractor,
        int nMode
);
```

Possible return values

UFE_OK, UFE_ERROR, UFE_ERR_INVALID_PARAMETERS, UFE_ERR_UNDEFINED_MODE

Parameters

hExtractor	[in] Handle of the extractor object
nMode	[in] Specifies the mode

See also

UFE_GetMode

Examples

Visual C++

```
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 {
        // 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
```

UFE_GetParameter

Gets parameter value.

```
UFE_STATUS UFE_API UFE_GetParameter(
        HUFExtractor hExtractor,
        int nParam,
        void* 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
```

UFE_SetParameter

Sets parameter value.

```
UFE_STATUS UFE_API UFE_SetParameter(
        HUFExtractor hExtractor,
        int nParam,
        void* 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++

```
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);
// 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
```

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
);
```

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
```

UFE_SetEncryptionKey

Sets encryption key.

```
UFE_STATUS UFE_API UFE_SetEncryptionKey(
        HUFExtractor hExtractor,
        unsigned char* 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++

```
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
}
```

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
```

UFE_EncryptTemplate

Encrypts template.

```
UFE_STATUS UFE_API UFE_EncryptTemplate(
        HUFExtractor hExtractor,
        unsigned char* pTemplateInput,
        int nTemplateSize,
        unsigned char* pTemplateOutput,
        int* 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
#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 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
```

UFE_DecryptTemplate

Decrypts template.

```
UFE_STATUS UFE_API UFE_DecryptTemplate(
        HUFExtractor hExtractor,
        unsigned char* pTemplateInput,
        int nTemplateSize,
        unsigned char* pTemplateOutput,
        int* 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
```

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
);
```

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 template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384
// Assume maximum possible image size is 480 x 640
#define MAX_IMAGE_WIDTH 480
#define MAX_IMAGE_HEIGHT 640

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 template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384
' Assume maximum possible image size is 480 x 640
Const MAX_IMAGE_WIDTH As Long = 480
Const MAX_IMAGE_HEIGHT As Long = 640

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
```

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
);
```

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 template size is 384 bytes
#define MAX_TEMPLATE_SIZE 384
// Assume maximum possible image size is 480 x 640
#define MAX_IMAGE_WIDTH 480
#define MAX_IMAGE_HEIGHT 640

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 template size is 384 bytes
Const MAX_TEMPLATE_SIZE As Long = 384
' Assume maximum possible image size is 480 x 640
Const MAX_IMAGE_WIDTH As Long = 480
Const MAX_IMAGE_HEIGHT As Long = 640

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
```

UFE_GetErrorString

Gets the error string for specified UFE_STAUS value.

```
UFE_STATUS UFE_API UFE_GetErrorString(
        UFE_STATUS res,
        char* 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 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
```

UFDatabase module

UFDatabase module provides functionality for managing database, adding / updating / removing / getting templates with user data, etc.

Requirements

Visual C++
• Required header: include\UFDatabase.h • Required lib: lib\UFDatabase.lib • Required dll: bin\UFDatabase.dll

Visual Basic 6.0
• 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)
Tempalte1	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=UFDatabase.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++

```
#define MAX_TEMPLATE_SIZE 1024
#define MAX_MEMO_SIZE 100
```

```
UFD_STATUS ufd_res;
HUFDatabase hDatabase;
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++

```
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
}
```

Visual Basic 6.0

```
Dim ufd_res As UFD_STATUS
Dim hDatabase As Long
Dim DataNumber As Long
```

```
' Get database handle, hDatabase

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 tempalte size list
pnSerial	[out] Receives the serial number list

See also

UFD_GetTemplateNumber

Examples

Visual C++

```
#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_GetTemplateNumber function

// Allocate templates list and serial list
ppTemplate = (unsigned char**)malloc(nTemplateNumber * sizeof(unsigned
```

```
char*));
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_STAUS 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 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 UniFinger Engine 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
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 of 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
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

SaveCaptureImageBufferToJPG	Saves the capture image buffer to the specified file of the jpeg format
ClearCaptureImageBuffer	Clears the capture image buffer
GetErrorString	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. Higher value means more sensitive. Default value is 4.

```
public int Sensitivity { 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 of 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(
        out Bitmap bitmap,
        out 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 of 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

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

UFM_STATUS	Every function in UFMatcher module returns UFM_STATUS enumeration value

Classes

UFMatcher	UFMatcher class provides functionality for verifying fingerprints using two templates, identifying fingerprints using the template array, etc

UFM_STATUS enumeration

Every function in UFMatcher 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_STAUS 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.

```
public enum UFE_STATUS : int
```

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
LoadImageFromTIFFile	Loads image data from tiff file
LoadImageFromJPGFile	Loads image data from jpeg file
LoadImageFromBMPBuffer	Loads image data from bitmap class instance
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

LoadImageFromTIFFile method

Loads image data from tiff file.

```
public UFE_STATUS LoadImageFromTIFFile(
        string TIFFileName,
        out byte[] ImageData,
        out int Width,
        out int Height
);
```

Possible return values

UFE_STATUS.OK, UFE_STATUS.ERROR,
UFE_STATUS.ERR_INVALID_PARAMETERS

Parameters

TIFFileName	[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

GetErrorString method

Gets the error string for specified UFE_STAUS 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

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

UFD_STATUS	Every function in UFExtractor module returns UFE_STATUS enumeration value

Classes

UFDatabase	UFDatabse 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

Open	Opens a database
Close	Closes specified database
AddData	Adds data into the specified database
UpdateDataByUserInfo	Updates the database entry having specified user ID and finger index
UpdateDataBySerial	Updates the database entry having specified serial number
RemoveDataByUserID	Removes the database entries having specified user ID
RemoveDataByUserInfo	Removes the database entries having specified user ID and finger index
RemoveDataBySerial	Removes the database entries having specified serial number
RemoveAllData	Removes all database entries
GetDataNumber	Gets the number of database entries
GetDataByIndex	Gets the database entry having specified index
GetDataByUserInfo	Gets the database entry having specified user ID and finger index
GetDataBySerial	Gets the database entry having specified serial number
GetTemplateListWithSerial	Gets all the template list with corresponding serial number list from specified database
GetErrorString	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_STAUS 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

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 UniFinger Engine SDK, send a mail to support@suprema.com or sales@suprema.com.

Appendix B. Distribution Content

UniFinger Engine SDK 3.0 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.dll	Scanner sub module for SFR200	PE, PS

[1] PE = UniFinger Engine SFR Enroll SDK, PS = UniFinger Engine SFR SDK, PM = UniFinger Engine Match SDK, PI = UniFinger Engine Image SDK

docs

File Name	Description	Supported Products [1)]
UFE30.chm	Reference Manual (Windows help file format)	PE, PS, PM, PI
UFE30.pdf	Reference Manual (Adobe PDF format)	PE, PS, PM, PI

[1)] PE = UniFinger Engine SFR Enroll SDK, PS = UniFinger Engine SFR SDK, PM = UniFinger Engine Match SDK, PI = UniFinger Engine 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 = UniFinger Engine SFR Enroll SDK, PS = UniFinger Engine SFR SDK, PM = UniFinger Engine Match SDK, PI = UniFinger Engine 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

[1)] PE = UniFinger Engine SFR Enroll SDK, PS = UniFinger Engine SFR SDK, PM = UniFinger Engine Match SDK, PI = UniFinger Engine 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 = UniFinger Engine SFR Enroll SDK, PS = UniFinger Engine SFR SDK, PM = UniFinger Engine Match SDK, PI = UniFinger Engine 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

[1)] PE = UniFinger Engine SFR Enroll SDK, PS = UniFinger Engine SFR SDK, PM = UniFinger Engine Match SDK, PI = UniFinger Engine Image SDK

Index

S

Suprema namespace 216, 265, 280, 299
Suprema.UFD_STATUS enumeration 300
Suprema.UFDatabase class 301
Suprema.UFDatabase module ..298
Suprema.UFDatabase.AddData method 304
Suprema.UFDatabase.Close method 303
Suprema.UFDatabase.GetDataByIndex method 312
Suprema.UFDatabase.GetDataBySerial method 314
Suprema.UFDatabase.GetDataByUserInfo method 313
Suprema.UFDatabase.GetDataNumber method 311
Suprema.UFDatabase.GetErrorString method 316
Suprema.UFDatabase.GetTemplateListWithSerial method....315
Suprema.UFDatabase.Open method 302
Suprema.UFDatabase.RemoveAllData method.....310
Suprema.UFDatabase.RemoveDataBySerial method 309
Suprema.UFDatabase.RemoveDataByUserID method.....307
Suprema.UFDatabase.RemoveDataByUserInfo method 308
Suprema.UFDatabase.UpdateDataBySerial method 306
Suprema.UFDatabase.UpdateDataByUserInfo method 305
Suprema.UFE_MODE enumeration 282
Suprema.UFE_STATUS enumeration 281
Suprema.UFExtractor class 283
Suprema.UFExtractor module ..279
Suprema.UFExtractor.DecryptTemplate method..... 292
Suprema.UFExtractor.DetectCore property..... 285
Suprema.UFExtractor.EncryptTemplate method..... 291
Suprema.UFExtractor.Extract method 288
Suprema.UFExtractor.GetErrorString method 297
Suprema.UFExtractor.LoadImageFromBMPBuffer method... 296
Suprema.UFExtractor.LoadImageFromBMPFile method 293
Suprema.UFExtractor.LoadImageFromJPGFile method..... 295
Suprema.UFExtractor.LoadImageFromTIFFile method..... 294
Suprema.UFExtractor.Mode property..... 284
Suprema.UFExtractor.SetEncryptionKey method..... 290
Suprema.UFExtractor.TemplateSize property 286
Suprema.UFExtractor.UseSIF property..... 287
Suprema.UFM_STATUS enumeration 266
Suprema.UFMatcher class..... 267
Suprema.UFMatcher module... 264
Suprema.UFMatcher.AbortIdentify method 274
Suprema.UFMatcher.FastMode property..... 268
Suprema.UFMatcher.GetErrorString method 278
Suprema.UFMatcher.Identify method 272
Suprema.UFMatcher.IdentifyInit method 275
Suprema.UFMatcher.IdentifyMT method 272

Suprema.UFMatcher.IdentifyNext method 276
Suprema.UFMatcher.RotateTemplate method 277
Suprema.UFMatcher.SecurityLevel property 269
Suprema.UFMatcher.UseSIF property.....270
Suprema.UFMatcher.Verify method 271
Suprema.UFS_CAPTURE_PROC delegate.....220
Suprema.UFS_SCANNER_PROC delegate.....219
Suprema.UFS_SCANNER_TYPE enumeration 218
Suprema.UFS_STATUS enumeration 217
Suprema.UFScanner class 233
Suprema.UFScanner module....215
Suprema.UFScanner.AbortCapturing method 252
Suprema.UFScanner.Brightness property.....238
Suprema.UFScanner.CaptureEvent event 235
Suprema.UFScanner.CaptureSingleImage method 250
Suprema.UFScanner.ClearCaptureImageBuffer method.....262
Suprema.UFScanner.DecryptTemplate method 256
Suprema.UFScanner.DetectCore property.....241
Suprema.UFScanner.DrawCaptureImageBuffer method.....258
Suprema.UFScanner.EncryptTemplate method 255
Suprema.UFScanner.Extract method 253
Suprema.UFScanner.GetCaptureImageBuffer method.....257
Suprema.UFScanner.GetErrorString method 263
Suprema.UFScanner.Handle property.....248
Suprema.UFScanner.ID property 236
Suprema.UFScanner.IsCapturing property..... 247
Suprema.UFScanner.IsFingerOn property..... 246
Suprema.UFScanner.IsSensorOn property..... 245
Suprema.UFScanner.SaveCaptureImageBufferToBMP method 259
Suprema.UFScanner.SaveCaptureImageBufferToJPG method 261
Suprema.UFScanner.SaveCaptureImageBufferToTIF method260
Suprema.UFScanner.ScannerType property..... 244
Suprema.UFScanner.Sensitivity property..... 239
Suprema.UFScanner.Serial property..... 240
Suprema.UFScanner.SetEncryptionKey method..... 254
Suprema.UFScanner.SetScanner method 249
Suprema.UFScanner.StartCapturing method 251
Suprema.UFScanner.TemplateSize property..... 242
Suprema.UFScanner.Timeout property..... 237
Suprema.UFScanner.UseSIF property..... 243
Suprema.UFScannerCaptureEventArgs class..... 222
Suprema.UFScannerManager class 223
Suprema.UFScannerManager.Init method 227
Suprema.UFScannerManager.ScannerEvent event 226
Suprema.UFScannerManager.ScannerList class 230
Suprema.UFScannerManager.ScannerList.Count property..... 231

Suprema.UFScannerManager.ScannerList.Item property 232
Suprema.UFScannerManager.Scanners property 225
Suprema.UFScannerManager.UFScannerManager constructor 224
Suprema.UFScannerManager.Uninit method 229
Suprema.UFScannerManager.Update method 228
Suprema.UFScannerManagerScannerEventArgs class 221

U

UFD_AddData 179
UFD_Close 177
UFD_GetDataByIndex 198
UFD_GetDataBySerial 204
UFD_GetDataByUserInfo 201
UFD_GetDataNumber 196
UFD_GetErrorString 212
UFD_GetTemplateListWithSerial 209
UFD_GetTemplateNumber 207
UFD_Open 175
UFD_RemoveAllData 194
UFD_RemoveDataBySerial 192
UFD_RemoveDataByUserID ...188
UFD_RemoveDataByUserInfo 190
UFD_UpdateDataBySerial 185
UFD_UpdateDataByUserInfo ..182
UFDatabase module 173
UFE_Create 146
UFE_DecryptTemplate 165
UFE_Delete 148
UFE_EncryptTemplate 163
UFE_Extract 158
UFE_GetErrorString 171
UFE_GetMode 150
UFE_GetParameter 154
UFE_LoadImageFromBMPBuffer 169
UFE_LoadImageFromBMPFile 167
UFE_SetEncryptionKey 161
UFE_SetMode 152
UFE_SetParameter 156
UFExtractor module 143
UFM_AbortIdentify 132
UFM_Create 117
UFM_Delete 119
UFM_GetErrorString 141
UFM_GetParameter 121
UFM_Identify 128
UFM_IdentifyInit 134
UFM_IdentifyMT 128
UFM_IdentifyNext 136
UFM_RotateTemplate 139
UFM_SetParameter 123
UFM_Verify 125
UFMatcher module 114
UFS_AbortCapturing 92
UFS_CaptureSingleImage 86
UFS_ClearCaptureImageBuffer 110
UFS_DecryptTemplate 100
UFS_DrawCaptureImageBuffer 106
UFS_EncryptTemplate 98
UFS_Extract 94
UFS_GetCaptureImageBuffer . 104
UFS_GetCaptureImageBufferInfo 102
UFS_GetErrorString 112
UFS_GetParameter 77
UFS_GetScannerHandle 67
UFS_GetScannerHandleByID ... 69
UFS_GetScannerID 73
UFS_GetScannerIndex 71
UFS_GetScannerNumber 65
UFS_GetScannerType 75
UFS_Init 59
UFS_IsCapturing 90
UFS_IsFingerOn 84
UFS_IsSensorOn 82
UFS_RemoveScannerCallback.. 64
UFS_SaveCaptureImageBufferToBMP 108
UFS_SetEncryptionKey 96
UFS_SetParameter 80
UFS_SetScannerCallback 62
UFS_StartCapturing 88
UFS_Uninit 61
UFS_Update 60

UFScanner module 54