UniFinger Engine .NET SDK Reference Manual

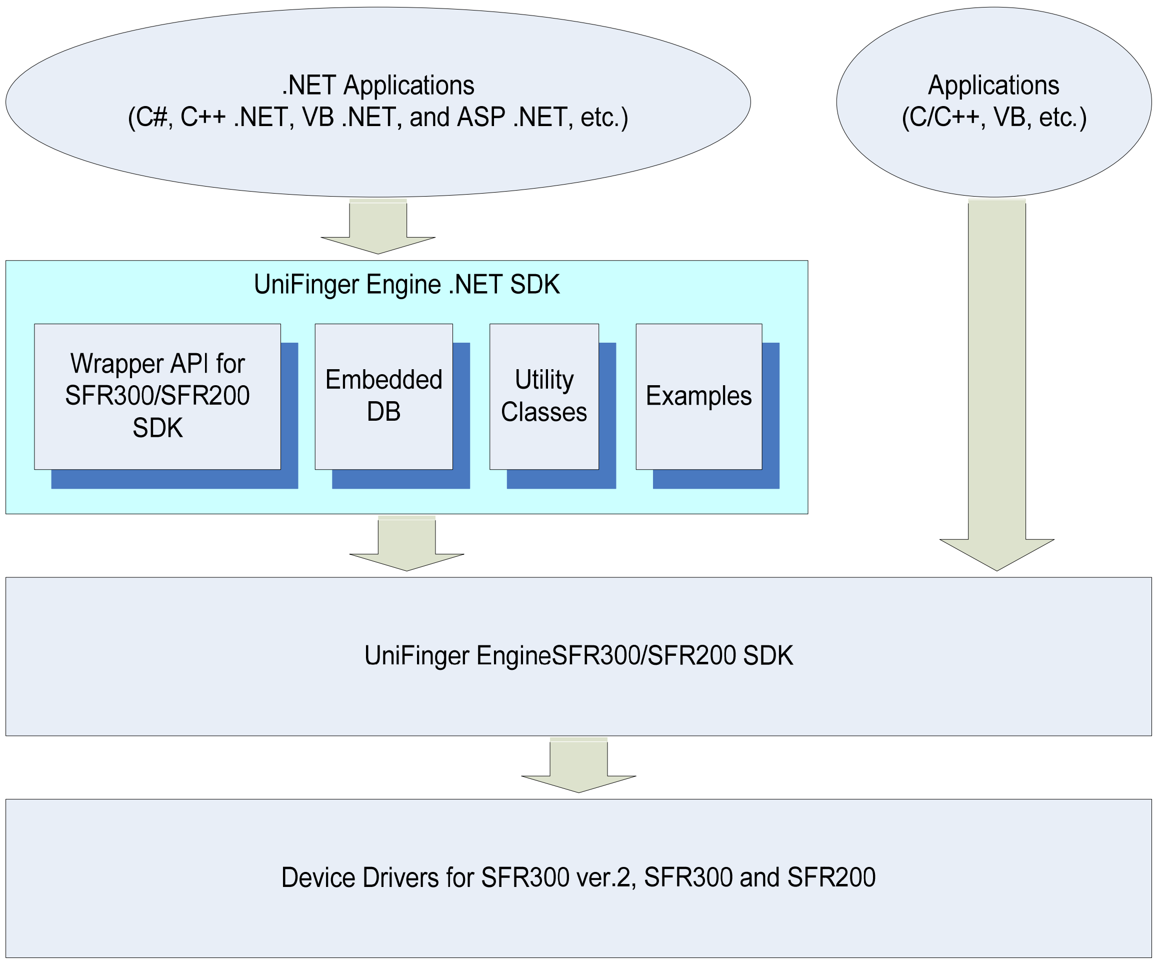
Version 2.6

SUPREMA

Contents

1. Introduction

UniFinger Engine .NET SDK provides developers with the tools for writing applications using SFR200 and SFR300 scanners in .NET languages. As shown in the following picture, the .NET SDK is placed on top of UniFinger Engine SFR300/SFR200 SDK, which are written in C++ and provided as native DLL. The SDK hides all the technical intricacies about calling the native DLL in .NET environments and provides well-defined .NET APIs. It also adds its own new functions such as database management and image manipulation. The source codes of an example application are also provided in C#, VB.NET, and ASP.NET.


.NET Applications
(C#, C++ .NET, VB .NET, and ASP .NET, etc.)
Applications
(C/C++, VB, etc.)
UniFinger Engine .NET SDK
Wrapper API for SFR300/SFR200 SDK
Embedded DB
Utility Classes
Examples
UniFinger EngineSFR300/SFR200 SDK
Device Drivers for SFR300 ver.2, SFR300 and SFR200

Features

- Using built-in DB

  Templates can be inserted, deleted, updated and selected without DB Connection string nor SQL string.

  ExportDB method transforms all the contents of a table into a file format(txt) and transfers it to another DB.
- Identification function

  Adding MatchTemplate, MatchTemlpaeDB method allows easier template matching.
- 23 properties are provided. Image quality can be acquired through Quality property and template image and template value generated at the end can be acquired through GetLastImage, GetLastTemplate method.
- Sample programs written in C#, VB.NET, ASP.NET with C# are provided for users to understand all the methods and properties of SDK.

Each chapter is made up as follows.

Chapter 2 Consists of system requirements and SDK installation.

Chapter 3 Consists of basic methods of using SDK: Initialization, environment setting, enrollment of finger prints, identification of finger prints and how to use built-in DB.

Chapter 4 Consists of methods and properties of SDK. Regarding important API, samples in C# and VB.NET are provided.

Customer Support

- Headquarters

  Suprema, Inc.(http://www.suprema.com)

  16F Parkview office Tower, Jeongja-dong, Bundang-gu, Seongnam, Gyeonggi, 463-863 Korea

  Tel: +82-31-783-4502

  Tel: +82-31-783-4503

- E-mail support

  If you wish to directly query a problem in the UniFinger Engine .NET SDK, send a mail to support@suprema.com, sales@suprema.com.

2. Installation and Structure

2.1. Development system requirements

Minimum system requirements are as follows.

- Install one out of .NET framework 1.0, 1.1, 2.0
- Pentium-class processor
- Windows 98SE, 2000 with service pack 4, Me, XP
- 32MB minimum physical RAM
- 10 MB minimum hard disk space during installation
- 5MB hard disk space for installation

2.2. SDK Structure

Once UniFinger Engine .NET SDK is installed,

- Three folders for sample programs written in C#, VB.NET, ASP.NET with C# are installed at DotNet₩TestProject
- Each sample program is written as to test all API of SDK
- Using Tool Tip allows users to check the sample programs.

	200		300 and 300 ver.2	
	File	Location	File	Location
Device	IZZIX.dll	Windows₩ System32	sfudusb.sys sfuusb.inf	Windows₩System32₩Drivers Windows₩Inf
DLL	SFR200.dll Suprema.dll	Windows₩ System32	SFR300.dll Suprema.dll	Windows₩System32

3. Using SDK

This chapter explains how to refer to and initialize SDK, and to enroll and identify finger prints in application program. Enrollment and identification of finger print differ considerably depending on whether or not built-in DB is used. In case of using built-in DB, a programmer can insert, delete or update a desired template without connection string to DB or SQL string. DB is a basic form provided, which cannot be expanded nor updated.

The procedures of fingerprint enrollment and identification are as follows.

- Initialize SDK for sensor use. This is a mandatory job in order to use a sensor.

- Environment Setting. Set basic properties for SDK use. If properties are not set, a default value used in SDK will be set for properties.

- Enroll fingerprint. Scan fingerprint for enrollment. A user places his/her finger on sensor and then the captured fingerprint is transmitted from sensor to PC. Fingerprint is generated in 384-byte array type and this fingerprint data is defined as a template.

- Fingerprint identification. Identification is to match two fingerprints and check if they are matched with each other. Obtain the fingerprint data (template) which a user wants to identify through fingerprint scan and get the identification result by matching the template and enrolled fingerprint data (template array).

The sample to explain process procedures has been written in C# as a basic language, and class has been declared as follows for the use of SDK class. The class name of all the following sample codes is sfr.

```
Suprema.SFR sfr = new Suprema.SFR();
```

3.1. Initialization

- Initialize for sensor use. Initialization is a mandatory job in order to use a sensor. After calling a method, information on the sensor in use can be acquired.

- Initialize for sensor use. Initialization is a mandatory job in order to use a sensor. After calling a method, information on the sensor in use can be acquired.
  ```
  bool returnvalue = sfr.Initialize();
  ```

- Get the index of a sensor in use with Device property. Index starts from 0.
  ```
  txtDevice.Text = sfr.Device.ToString();
  ```

- Get the number of a sensor currently connected to a system with number.
  ```
  txtDNumber.Text = sfr.DeviceNumber.ToString();
  ```

- Get a serial number of a sensor in use with Device Serial property
  ```
  txtSerial.Text = sfr.DeviceSerial.ToString();
  ```

- When terminating application program, call Uninitialize in order to release memory in use. Released memory is not the one related to application program but the one used for SDK

3.2. Environment Setting

Basic property values, which are required to use SDK, are set. If the values are not set, default values of SDK automatically become basic property values.

3.2.1. Sensor Setting

Set the sensor to be used in case SFR300-S ver.2, SFR300-S and SFR200-S are installed at the same time. If a sensor is not set, SFR300 ver.2 is set as default. In the event that either SFR300-S ver.2, SFR300-S or SFR200-S is set, sensor setting is not required. Available type of a sensor can be selected from SFR class data type, or SensorType.

```
sfr.Sensor = Suprema.SFR.SensorType.SFR300V2;
```

3.2.2. Security Level Setting

Set the security level for fingerprint identification. Security level refers to FAR (False Accept Rate). For instance, Level 4 indicates that one out of 100000 attempts to enroll fingerprints can be false. Default is 4. FAR value which is security level is as follows.

Security Level	False Accept Rate(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)

3.2.3. Image Quality Setting

Set a critical value of fingerprint image quality to be scanned. Quality of fingerprint image depends on the size, noises and brightness of scanned fingerprint. Therefore, quality of fingerprint image is not quality of scanned fingerprint itself (template) but quality of fingerprint image. In case the value of fingerprint image quality is below the critical value, an error will be returned. The value ranges from 300 to 100 and default is 70. This property can be also performed in method such as ScanTemplate, ScanRotateTemple, by setting qualityCheck parameter as true.

3.2.4. Match Mode Setting

Match Mode determines the speed of matching process in the event of fingerprint identification. Through a data type called MatchType, users can select either Normal-Mode or Fast_Mode to determine the speed. Selecting Fast_Mode enables the identification process to speed up.

```
Sfr.MatchMode = Suprema.SFR.MatchType.FAST_MODE;
```

3.2.5. Timeout Setting

Set standby time at a time of fingerprint identification. In case of failure to identify within a set time, an error is returned. Value ranges from 0 to 10 seconds and default is 5 second. If it is set as 0 second, identifications continue without time limit.

3.2.6. Fingerprints Brightness Setting

Set brightness of fingerprint. The value ranges from 0 to 255 and default is 100 in SFR300 SDK and default is 50 in SFR200 SDK. The greater the value becomes, the darker the image gets.

3.2.7. Sensor Sensitivity Setting

Sensor Sensitivity means how sensitively a sensor detects a finger. The higher sensor sensitivity gets, the more easily fingerprint is accepted whereas the lower it is, the stabler fingerprint can be scanned. The value ranges from 0 to 7 and default is 4.

3.3. Enrollment and Identification of Fingerprints

Unifinger Engine for .NET SDK 2.5 supports built-in DB for administration of fingerprint data. The procedures of fingerprint enrollment and identification differ depending on whether or not built-in DB is used.

3.3.1. In case built-in DB is not used

Template which is acquired after scanning fingerprint is operated according to development environment of a programmer. At a time of identification, either scan a desired template for identification or match an existing template with an object template.

3.3.1.1. Scanning Fingerprints

A user who wants to scan his fingerprints puts his finger on a sensor to obtain a fingerprint template according to the parameter which was previously set. This fingerprint template takes the form of 384-byte array type. In the process of scanning, Two Template Mode is provided to increase recognition rate of fingerprint.

byte[] ScanTemplate(bool coreDetect, bool qulityCheck)

ScanTemplate Method scans fingerprints from the sensor and returns them in the form of 384-byte array type. The first parameter detects a core of the fingerprint. If it fails to detect it from the center of the sensor, it returns Scan.Error. The second parameter checks image quality. In case of True, it makes a comparison between obtained image of the fingerprint and ImageCheckQuality which was previously set in the Environment Setting stage. If the image quality of the obtained fingerprint is poorer, the method returns Error.

Two template mode is a factor which has each finger scanned twice. Two template mode is recommended as it can increase recognition rate of fingerprint. In case of using a built-in DB, this mode is supported automatically by setting a parameter, but if not, a programmer needs to set this mode. If the first scanned template matches with the rescanned template, both scanned templates are used. If the same finger is placed continuously without being removed before the second scanning, the sensor just reads the previous fingerprint image twice. In this case, call IsFingerOn method to obtain another template of the same fingerprint.

- The following codes show how templates are created in two template mode.

```
// declare template
byte[] temptemplate1 = new byte[384];
byte[] temptemplate2 = new byte[384];

// create template
temptemplate1 = sfr.ScanTemplate(chkCore.Checked, chkQuality.Checked);
while (sfr.IsFingerOn())
{
        MessageBox.Show("Put Off Your Finger.", "Two Template Warning",
                              MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
temptemplate2 = sfr.ScanTemplate(chkCore.Checked, chkQuality.Checked);

//match template
if (sfr.MatchTemplate(temptemplate1, temptemplate2))
        txtQ2.Text = sfr.Quality.ToString();
else
        MessageBox.Show("Not Match Template.", "Two Template Error",
                                        MessageBoxButtons.OK, MessageBoxIcon.Error);
```

- The following codes show how several template data are inserted into array variable. These templates can be used for 1:N identification at a time of template identification. 1:N identification will be further explained in the section on identification.

```
IntPtr[] templateDB = new IntPtr[10];
```

```
 int templateIndex = 0;
temptemplate1 = sfr.ScanTemplate(chkCore.Checked, chkQuality.Checked);
// allot memory
for( int i = 0; i < 10; i++ )
{
    templateDB[i] = Marshal.AllocHGlobal( 384 );
}
 //copy template's address in templateDB array
 Marshal.Copy( temptemplate1, 0, templateDB[templateIndex++], 384 );
```

- Refer to Chapter 3 for code and message which are returned in case of Scan Failure.

3.3.1.2. Image and Image Quality

- Call GetLastImage to obtain the last scanned image data

```
pic1.Image = sfr.GetLastImage();
```

- Use Quality property to obtain the last scanned image quality.

```
txtQ.Text = sfr.Quality.ToString();
```

3.3.1.3. Identification

Identification is a process to compare two templates and check if they are matched with each other. There are two ways to identify templates: 1:1 matching and 1:N matching. In the former process, templates are compared and matched one-on-one, while the latter process finds two matched templates out of multiple templates. For further explanation, refer to a sample of MatchTemplate method in chapter 3.

```
//1:1 Matching
bool MatchTemplate(byte[] sourcetemplate, byte[] currenttemplate)

//1:N Matching
bool MatchTemplate(IntPtr[] templateDB, int tcount,
                                        byte[] currenttemplate, out int index)
```

- 1:1 Matching

  Validate success or failure of identification by using MatchTemplate. For 1:1 matching, matching can be done by setting desired data out of those saved in scan, file or DB at 384-byte array variable.

  Next, copy 0th template saved in templateDB variable as 1:1 matching code to bytes variable, then match it with temptemplate1 template. Creation of templateDB is the above fingerprint scan code part.

```
byte[] bytes = new byte[384];

 // Copy the value of templateDB 0 in bytes variable
 Marshal.Copy(templateDB[0], bytes, 0, 384 );

 // match with created temptemplate1
 bool returnvalue = sfr.MatchTemplate(temptemplate1, bytes);
```

- 1:N Matching

  In case of 1:N matching, set the addresses of templates saved in user DB or file using array variable of IntPtr type, and match with desired template for identification. Identification conditions can be established by setting properties of security level and timeout. As for these 2 properties, refer to the section on Environment Setting. When matching multiple templates with one template, it is recommended to call 1:N matching method than to keep calling 1:1 matching method with a loop in view of speed. Next, transfer array index of templates which match with 1:N matching code to reference variable and the result of whether or not identification has been successful.

```
sfr.MatchTemplate(templateDB, templateIndex-1, temptemplate1, out index);
txtFingerIdx.Text = index.ToString();
```

- For further information on MatchTemplate, refer to Reference in the chapter 3.

3.3.1.4. Return Massage

- After any method is called, the resulting code value and message are shown through ReturnCode and ReturnMessage properties.
- Integer-type ReturnCode returns 1 in case of success, and returns Error Code in case of failure. Refer to Appendix A for detailed information on Error Code.
- String-type ReturnMessage delivers messages according to above ReturnCode value.

- The following is an example showing that ReturnCode and ReturnMessage are indicated in status bar.

```
this.statusBarPanel1.Text = "Message : " + sfr.ReturnMessage;
this.statusBarPanel2.Text = "Code : "     + sfr.ReturnCode.ToString();
```

3.3.2. Built-in DB

Insert template data by using provided access DB (mdb). Programmer can insert, delete, update or select data by using given method without writing DB connection string or SQL.

Procedures of enrolling and verifying fingerprint by using built-in DB are as follows.

- Create DB file and table to be used with CreateDatabase method.
- Get a template to be inserted by using ScanTemplate method.
- Execute fingerprint enrollment process to insert a template by using InsertTemplate method
- The above 2 processes can be done through EnrollTemplate method.
- Execute identification process to find a matched template using MatchTemplateDB method.

3.3.2.1. Structure of DB

The name of DB is enroll.mdb and the name of table is FingerPrints.
UserID is a primary key of a table and a string type of no more than 10 bytes. FingerIndex is an index of an applicable finger. Ten fingers can be enrolled per 1 UserID, and the integer type can be set by the programmer. Numbers are given to each finger, which makes it easy to distinguish each finger. The designated numbers cannot be duplicated in one UserID. Memo field can be used at the disposal of the programmer. Fingerprint1 and Fingerprint2 are template data, and Fingerprint1 is required to be entered. With Two Template Mode disabled, Null value is saved for Fingerprint2.

- Table Structure

Field Name	Data type	Description	Declaration in application program
UserID	String/10byte	Primary Key	String
FingerIndex	Int	Fingerprint Index	
Memo	String/50byte		String
Fingerprint1	OLE object	Mandatory input	Byte[] template1 =

		Of template contents	new byte[384]
Fingerprint2	OLE object	Template contents	Byte[] template2 = new byte[384]

3.3.2.2. Create Data Table

In case of using built-in DB for the first time, create a DB file of Enroll.mdb and FingerPrint Table by using CreateDatabase method. The DB file is located in route directory where application program is operated. But in ASP.NET, designated location is found by WEB.CONFIG. Below shows connection string array info set in web.config and this is designated by a programmer.

```
<appSettings>
<add key="connectionstring"
            value="Provider=Microsoft.Jet.OLEDB.4.0;Data
            Source=C:₩suprema₩SFRExampleASPNET₩enroll.mdb; "/>
</appSettings>
```

3.3.2.3. Enroll with EnrollTemplate Method

Save scanned template directly in DB.

Public bool EnrollTemplate(string userID, int fingerIndex, string memo, bool twoTemplateMode, bool rotate, bool coreDetect, bool qulityCheck)

Through EnrollTemplate Method, templates scanned from the sensor can be saved directly in DB table. The fourth parameter is Two Template Mode, a recommended factor that has one finger scanned twice, increasing an identification rate of fingerprints. If two templates scanned twice are not matched, an error is returned. The fifth parameter rotates fingerprints. This parameter is used only when scanned fingerprint templates should be rotated 180 degrees. The next code saves UserID 120, FingerIndex 0 and memo Test and templates to be scanned in DB. This code detects the core of fingerprints and checks image quality but does not operate Two Template

Mode nor rotate.

```
Bool returnValue = sfr.EnrollTemplte(“120”,   0, “Test”, true, false, true, true)
```

- After being inserted in DB, inserted template, image and image quality can be acquired. Able to get the template created at the end by using GetLastTemplate method.

```
pic1.Image = sfr.GetLastImage();
byte[] template = sfr.GetLastTemplate();
txtQ.Text = sfr.Quality.ToString();
```

3.3.2.4. Insert Data with InsertTemplate

After creating a template to be inserted, create user information and call InsertTemplate method.

bool InsertTemplate(string userid, int index, string memo, byte[] template1, byte[] template2)

InsertTemplate method is available only for insertion where as EnrollTemplate method enables scan and insert at the same time. If required entry field is Null value, it is not saved and an error value is returned, and if it’s not TwoTemplate mode, Null value transfers template2. The below is InsertTemplate code and temptemplate2 value is Null.

```
byte[] temptemplate1 = new byte[384];
byte[] temptemplate2 = new byte[384];

temptemplate1 = sfr.ScanTemplate(chkCore.Checked,
                                                chkQuality.Checked);
bool rvalue = sfr.InsertTemplate(txtUserId.Text,
Convert.ToInt32(txtIndex.Text), txtMemo.Text, temptemplate1,
                                                      temptemplate2);
```

3.3.2.5. Identification

Identification is a process to match the template inserted in built-in DB with a subject template that needs to be matched and to return user information of a matched template.

Call MatchTemplateDB method. In case of success, applicable UserID and user information are returned in dataRow type, while in case of failure, null dataRow is returned. The below is the code matching created template and DB values.

```
DataRow dr = sfr.MatchTemplateDB(temptemplate1);
if (sfr.ReturnCode == 1)
{
        txtUserId.Text = dr[0].ToString();
        txtIndex.Text   = dr[1].ToString();
}
```

3.3.2.6. Select Data

This method is used to search data table contents. Call SelectTemplate method without writing SQL. Return value is DataTable type.

- Bring the entire data

  ```
  DataTable dTable = sfr.SelectTemplate();
  ```

- Bring data on UserID

  ```
  DataTable datatable = sfr.SelectTemplate(txtUserId.Text);
  ```

- Bring data between UserIDs

  ```
  DataTable datatable = sfr.SelectTemplate(txtUserId.Text,
                                            txtTouser.Text);
  ```

- Bring FingerIndex for UserID and connect it to datagrid.

  ```
  dataGrid1.DataSource =
          sfr.SelectTemplate(txtUserId.Text, Convert.ToInt32(txtIndex.Text));
  dataGrid1.Refresh();
  ```

3.3.2.7. Delete Data

This method is used to delete inserted templates. To use, call DeleteTemplate Method.

- Delete the entire data on the UserID
  bool rvalue = sfr.DeleteTemplate(txtUserId.Text);

- Delete data on specific fingerprint index of the UserID
  bool rvalue = sfr.DeleteTemplate(txtUserId.Text, txtIndex.Text);

3.3.2.8. UpdateData

This method is used to update inserted templates or memos. UserId and FingerIndex field, which are a primary key values, cannot be updated. Call UpdateTemplate Method to use this method.

- Update Template1 and Template2 of UserID
  bool rvalue = sfr.UpdateTemplate(txtUserId.Text, index, template1, template2);

- Update memo on UserID
  bool rvalue = sfr.UpdateTemplate(txtUserId.Text, index , memo.Text);

- Update Template1, Template2 and memo of UserID
  bool rvalue = sfr.UpdateTemplate(txtUserId.Text, index, memo.Text, template1, template2);

4. SDK Reference

This chapter explains definition of SDK class SFR, method and property defined in SFR class and message as per error code.

4.1. SFR Class

SFR class which uses Suprema as name space consists of 24 methods and 23 properties.
Available 24 methods are as follows.

Classification	Name of method	Use of built-in DB	Not use of built-in DB
Device	Initialize	O	O
	UnInitialize	O	O
	IsSensorOn	O	O
	IsFingerOn	O	O
Scan	ScanTemplate	O	O
	ScanRotateTemplate	O	O
	EnrollTemplate	O	X
	ScanImage	O	O
	AbortScan	O	O
	GetLastTemplate	O	O
	GetLastImage	O	O
Match	MatchTemplate	O	O
	MatchTemplateMT	O	O
	MatchTemplateDB	O	X
	MatchTemplateDBMT	O	X
	AbortMatch	O	O
Database	CreateDatabase	O	X
	SelectTemplate	O	X
	InsertTemplate	O	X
	DeleteTemplate	O	X
	UpdateTemplate	O	X

	ExportDB	O	X
Image	AdjustImage	O	O
	SaveImageFile	O	O
	ReadFileToBitmap	O	O

4.2. Device

Methods for basic jobs to use a sensor

- Initialize method initializes a sensor, grasps a type of a sensor and check its license.
- Uninitialize method releases memory used in SFR class.
- IsSensorOn checks if sensor is connected to system.
- IsFingerOn checks if a finger is placed on a sensor. This method can be used to check whether or not a finger is contacted to get another fingerprint template of the same finger when using TwpTemplate Mode.

Out of these methods, Initialize method is a required one for using a sensor and Uninitialize method must be called to release memory used in SFR class before application program is closed.

Initialize

This is a method implemented to initialize the environment before using a sensor. It confirms license and inspects condition of a sensor. It is a mandatory method for using a sensor.

public bool Initialize();

Parameters

No parameters

Return value

	ReturnCode	ReturnMessage
Success	1	Initialize Success
Failure	0	Initialize Fail

Example

C#

```
private void frmSFRTest_Load(object sender, System.EventArgs e)
{
        // Initialize
        bool returnvalue = sfr.Initialize();
}
```

VB.NET

```
Private Sub frmSFR_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Dim rvalue As Boolean
        rvalue = sfr.Initialize()
End Sub
```

Uninitialize

This is a method implemented to stop the operation of a sensor by releasing memory used in SFR object.

public void UnInitialize();

Parameters

No parameters

Return value

No pass value

	ReturnCode	ReturnMessage
Success		
Failure		

Example

IsSensorOn

Check if UniFinger sensor is well connected.

public bool IsSensorOn()

Parameters

No parameters

Return value

Return 1 if a sensor is detected, and otherwise, return 0

	ReturnCode	ReturnMessage
Success	1	Sensor is Good
Failure	0	Please Check your Sensor

Example

C#

```
If (sfr.IsSensorOn())
 {
 }
 else
           this.statusBarPanel1.Text = "Message : " + sfr.ReturnMessage;
```

VB.NET

```
Dim returnvalue as bool
returnvalue = sfr.IsSensorOn()
```

IsFingerOn

Check if a finger is placed on UniFinger sensor. In case of using TwoTemplate Mode, this is used to check whether or not a finger was once removed and then placed again to get another type of fingerprint.

public bool IsFingerOn()

Parameters

No parameters

Return value

Returns True if a finger is placed on a sensor, and otherwise returns False.

	ReturnCode	ReturnMessage
Success	1	
Failure	0	Put on your Finger

Example

C#

```
while (sfr.IsFingerOn())
{
        MessageBox.Show("Put Off Your Finger.", "Two Template Warning",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
temptemplate2 = ScanTemplate(true, false);
```

VB.NET

```
While (sfr.IsFingerOn())
     MessageBox.Show("Put Off your Fingger", " Two Template Warning",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning)
End While
temptemplate2 = ScanTemplate(False, False)
```

4.3. Scan

Scan is to capture fingerprint data on sensor and to return features of captured fingerprint data in byte array type.

- Through ScanTemplate method, template of captured fingerprint data is created in 384 byte array.
- ScanRotateTemplate rotates templates 180 degrees, if necessary, according to the sensor.
- EnrollTemplate is the method which combines ScanTemplate method and the following InsertTemplate. It inserts scanned template to built-in DB immediately.
- When a user wants to get a fingerprint image after calling the above 3 methods, the fingerprint image created most recently can be obtained if GetLastImage method is called.
- ScanImage method captures fingerprint data and return fingerprint image only. When a user wants to get a fingerprint template after calling this method, call GetLastTemplate method and then a fingerprint template can be obtained.
- AbortScan method, whose method is to abort scanning process, is used in muti-thread environment.

ScanTemplate

Return fingerprint data which is scanned from a sensor in 384 byte array type.

public byte[] ScanTemplate(bool coreDetect, bool qulityCheck)

Parameters

coreDetect

Detects a core of a fingerprint. In case of True, if the core of the fingerprint is not detected, an error value is returned.

qulityCheck

Checks image quality. In case of True, image quality which is poorer than ImageCheckQuality property values is obtained, Scan.Error is returned.

Return value

If scan is successful, 384 byte array is returned. If not, byte array filled with null is returned.

	ReturnCode	ReturnMessage
Success	1	Scan is successful
Failure	0	Image is not good Core is not detected in the center of the image Scan Fail Please Check your Sensor Put on your Finger

Example

TwoTemplate Mode which increases identification rate by capturing 2 fingerprints per 1 finger is not supported automatically. Therefore, a user, who wants to get 2 templates for 1 finger as in TwoTemplate Mode, codes as per the below sample.

C#

```
byte[] temptemplate1 = new byte[384];
byte[] temptemplate2 = new byte[384];

// Get the first fingerprint of a finger
```

```
temptemplate1 = sfr.ScanTemplate(true, true);
pic1.Image = sfr.GetLastImage();
txtQ.Text = sfr.Quality.ToString();

// Remove a finger and place it again to get the second fingerprint of the same finger
while (sfr.IsFingerOn())
{
        MessageBox.Show("Put Off Your Finger.", "Two Template Warning",

MessageBoxButtons.OK,MessageBoxIcon.Warning);
}

// Get the second fingerprint of the same finger
temptemplate2 = sfr.ScanTemplate(true, true);
if (sfr.ReturnCode == 1)
{
        // Confirm if it's the fingerprint of the same finger
        if (sfr.MatchTemplate(temptemplate1, temptemplate2))
        {
                pic2.Image = sfr.GetLastImage();
                txtQ2.Text = sfr.Quality.ToString();
        }
        else
                MessageBox.Show("Not Match Template.", "Two Template Error",
                                MessageBoxButtons.OK,MessageBoxIcon.Error);
}
```

VB.NET

```
Dim temptemplate1(384) As Byte
Dim temptemplate2(384) As Byte

temptemplate1 = sfr.ScanTemplate(True, False)

If sfr.ReturnCode = 1 Then
        pic1.Image = sfr.GetLastImage()
        txtQ.Text = sfr.Quality.ToString()

        If (chkTwo.Checked) Then
            While (sfr.IsFingerOn())
                MessageBox.Show("Put Off your Fingger", "Two Template
                Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            End While

temptemplate2 = sfr.ScanTemplate(True, False)

If (sfr.ReturnCode = 1) Then
                If (sfr.MatchTemplate(temptemplate1, temptemplate2)) Then
                    pic2.Image = sfr.GetLastImage()
                    txtQ2.Text = sfr.Quality.ToString()
                Else
                    MessageBox.Show("Not Match Template.", "Two Template
                        Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                End If
            End If
        End If
End If
```

ScanRotateTemplate

Some sensors rotate templates 180 degrees. They rotate fingerprint template data 180 degrees to return them in byte array type.

public byte[] ScanRotateTemplate(bool coreDetect, bool qulityCheck)

Parameters

coreDetect

Detects a core of a fingerprint. In case of True, if the core of the fingerprint is not detected, an error value is returned.

qulityCheck

Checks image quality. In case of True, image quality which is poorer than ImageCheckQuality property values is obtained, Scan.Error is returned.

.

Return value

Success returns True, and Failure returns False.

	ReturnCode	ReturnMessage
Success	1	Scan is successful
failure	0	Image is not good Core is not detected in the center of the image Scan Fail Please Check your Sensor Put on your Finger

Example

EnrollTemplate

Insert fingerprint template scanned through sensor and user information to built-in DB. The method is available only when using built-in DB, which combines ScanTemplatedml and InsertTemplate described in the section on Database. As for database, refer to CreateDatabase section in the later part. In order to get fingerprint template and image after scanning, call GetLastTemplat, GetLastImage method.

public bool EnrollTemplate(string userId, int index, string memo, bool twoTemplate, bool rotate, bool coreDetect, bool qulityCheck)

Parameters

twoTemplate
To have a finger scanned twice. It is recommended as it creates another type of a template for 1 finger and increases identification rate of a fingerprint. If 2 scanned templates are not matched with each other, an error value is returned.
rotate
Set to decide whether to rotate fingerprint. This method is set to rotate scanned fingerprint template 180 degrees.
coreDetect
Decides whether to detect a core of fingerprint. In case of True, if the core of fingerprint is not detected from the center of the sensor, an error value is returned.
qulityCheck
Checks image quality. In case of True, image quality, which is poorer than ImageCheckQuality property values set in the Environment Setting stage, is obtained, Scan.Error is returned.

Return value

Success returns True, and Failure returns False

	ReturnCode	ReturnMessage
Success	1	Template is successfully inserted.
Failure	0	Image is not good Core is not detected in the center of the image Scan Fail

		Please Check your Sensor Put on your Finger Cannot find a table Check Data type Exceeds the maximum inserted number(10) of UserID Fail to connect to DB Fail to insert

Example

Insert 120 UserIDs, memo Test and template to be scanned into DB. Each finger is scanned twice, fingerprint data does not rotate, the core of the fingerprint is detected and image quality is checked. With ScanImage method, set the last template image created in the PIctureBox..

C#

```
if (txtUserId.Text.Length <= 0)
        MessageBox.Show("Input UserID", "Enroll", MessageBoxButtons.OK,
                                        MessageBoxIcon.Error);
else
{
        if (txtIndex.Text.Length <= 0 )
                MessageBox.Show("Input Finger Index", "Enroll",
                                MessageBoxButtons.OK,MessageBoxIcon.Error);
        else
        {
                MessageBox.Show("Put On your Fingger", "Enroll",
                        MessageBoxButtons.OK,MessageBoxIcon.Information);

                sfr.EnrollTemplate(txtUserId.Text, Convert.ToInt32(txtIndex.Text),
        txtMemo.Text, true, false,true, true);
                pic4.Image = sfr.ScanImage();
        }
}
```

VB.NET

```
        If (txtUserId.Text.Length <= 0) Then
                MessageBox.Show("Input UserID", "Enroll", MessageBoxButtons.OK,
                                                                        MessageBoxIcon.Error)
        Else
            If (txtIndex.Text.Length <= 0) Then
                 MessageBox.Show("Input Finger Index", "Enroll", MessageBoxButtons.OK,
                                                                        MessageBoxIcon.Error)
            Else
                 MessageBox.Show("Put On your Fingger", "Enroll",
                                MessageBoxButtons.OK, MessageBoxIcon.Information)

                sfr.EnrollTemplate(txtUserId.Text, Convert.ToInt32(txtIndex.Text),
                txtMemo.Text, chkTwo1.Checked, chkRotate.Checked, chkCore1.Checked,
                chkQuality1.Checked)

                pic4.Image = sfr.ScanImage()
             End If
        End If
```

ScanImage

Return fingerprint image scanned from the sensor in bitmap type.

public Bitmap ScanImage()

Parameters

No parameters.

Return value

Success returns bitmap, and failure returns null.

	ReturnCode	ReturnMessage
Success	1	Scan is successful
Failure	0	Scan Fails

Example

C#

```
pic1.Image = sfr.ScanImage();
```

VB.NET

```
pic1.Image = sfr.ScanImage()
```

AbortScan

This method, used to abort scanning process, is performed in the multi-thread environment.

public void AbortScan()

Parameters

No parameters

Return value

No return value

	ReturnCode	ReturnMessage
Success	1	
Failure	0	

Example

GetLastTemplate

Return the last created template. Fingerprint template can be obtained through this method after call of ScanImage method.

public byte[] GetLastTemplate ()

Parameters

No parameters

Return value

Success returns 384 byte array, and failure returns Null array.

	ReturnCode	ReturnMessage
Success	1	
Failure	0	No tempate

Example

C#

```
pic1.Image = sfr.ScanImage();
Byte[] template = sfr.GetLastTemplate ();
```

VB.NET

```
pic1.Image = sfr.ScanImage();
Dim template As Byte()
template = sfr.GetLastTemplate ()
```

GetLastImage

Return the last created image in buffer.

public Bitmap GetLastImage ()

Parameters

No parameters

Return value

Success returns bitmap and failure returns Null value.

	ReturnCode	ReturnMessage
Success	1	
failure	0	No templates are created.

Example

C#

```
 Bitmap image = sfr.GetLastImage ();
pic1.Image = image;
```

VB.NET

```
Dim image As Bitmap
image = sfr.GetLastImage ()
pic1.Image = image
```

4.4. Match

Match is a process to identify templates. There are two ways to proceed Match: 1:1 matching method and 1:N matching method. The former matches two templates one-on-one, and the latter searches two matched templates out of multiple templates.

● MatchTemplate method is used as overloaded with 4 ways, namely, matching a fingerprint to be scanned with 1 fingerprint template which has been already created or with multiple fingerprint templates, and matching created template with 1 template of fingerprint to be scanned or array of already created templates, according to transferring parameters. When matching multiple templates with 1 template, it is recommended to use 1:N matching rather than 1:1 matching to speed up the matching process.

● MatchTemplateDB method matches template with the one inserted in built-in DB and then return matched user information in DataRow type. It is used as overloaded with 2 ways, namely, matching with created template and comparing template to be scanned.

● AbortMatch method is used to abort matching in process. It can be used to stop matching when matching continues unlimitedly by setting Timeout property as 0.

MatchTemplate

Match templates and return the result. 1:1 matching and 1:N matching are available. In case of 1:N matching, it is recommended to use 1:N matching rather than 1:1 matching with a loop to boost its speed.

public bool MatchTemplate(parameters)

	parameter	Description
1:1	byte[] template, bool coreDetect, bool qulityCheck	Match created template with template to be scanned
	byte[] template1, byte[] template2	Match two created templates
1:N	IntPtr[] templateDB, int tcount, bool coreDetect, bool qulityCheck, out int index	Match multiple templates with template to be scanned. Return array index of templates which is matched with reference variable index.
	IntPtr[] templateDB, int tcount, byte[] currenttemplate, out int index	Match multiple templates and created template. Return array index of templates which is matched with reference variable index.

Parameters

template

Original template to be matched

templateDB

An array of original template to be matched. This sets starting address of each template as factor value.

coreDdetect

Detect the core of the fingerprint. In case of True, if the core of the fingerprint is not detected, an error value is returned.

qulityCheck

Check image quality. In case of True, if obtained image quality is poorer than ImageCheckQuality set in the Environment Setting stage, scan fails.

Return value

Success returns True, and failure returns False.

	ReturnCode	ReturnMessage
Success	1	Success Match
failure	0	Match Fail Matching template is NULL Template to be matched is NULL TimeOut

Example

1:1 matching

Match temporary template (temptemplate1) and scan template, then get scanned template and image as reference variable and return the result of whether they are matched or not.

C#

```
        byte[] temptemplate1 = new byte[384];

        temptemplate1 = sfr.ScanTemplate(true, true);
        bool returnvalue = sfr.MatchTemplate(temptemplate1, true, true);

        if (sfr.ReturnCode == 1)
        {
                pic3.Image = sfr.GetLastImage();
                txtQ.Text = sfr.Quality.ToString();
                }
```

VB.NET

```
        Dim returnvalue As Boolean
        Dim temptemplate1(384) As Byte

        temptemplate1 = sfr.ScanTemplate(True, True);
        returnvalue = sfr.MatchTemplate(temptemplate1, True, True)

        If sfr.ReturnCode = 1 Then
```

```
                pic3.Image = sfr.GetLastImage()
                txtQ.Text = sfr.Quality.ToString()
        End If
```

1:N matching

Return array index of template which is matched with index.

C#

```
   IntPtr[] templateDB = new IntPtr[10];
   int templateIndex = 0;
   byte[] template = new byte[384];

   // allot memory in templateDB
   for( int i = 0; i < 10; i++ )
        templateDB[i] = Marshal.AllocHGlobal( 384 );

  // create templateDB with 10 templates
  for( int i = 0; i < 10; i++ )
  {
        template = sfr.ScanTemplate(true, true);
        Marshal.Copy( template, 0, templateDB[templateIndex++], 384 );
  }

  //create template to be matched
  byte[] temptemplate1 = new byte[384];
  temptemplate1 = sfr.ScanTemplate(true, true);

  // match templates
  sfr.MatchTemplate(templateDB, templateIndex-1, temptemplate1, out index);
  txtFingerIdx.Text = index.ToString();
```

VB.NET

```
        Dim templateDB(10) As IntPtr
        Dim templateIndex As Integer = 0
        Dim template(384) As Byte
        Dim I As Integer
```

```
// allot memory in templateDB
For i = 0 To 10
        templateDB(i) = Marshal.AllocHGlobal(384)
Next i

// create templateDB with 10 templates
For i = 0 To 10
        template = sfr.ScanTemplate(true, true);
        Marshal.Copy(template, 0, templateDB(templateIndex), 384)
templateIndex += 1
Next i

//create template to be matched
Dim temptemplate1 (384) As Byte
temptemplate1 = sfr.ScanTemplate(true, true)

// match templates
sfr.MatchTemplate(templateDB, templateIndex - 1, temptemplate1, index)
txtFingerIdx.Text = index.ToString()
```

MatchTemplateMT

MatchTemplateMT method used by users with Dual CPU, has the same method as MatchTemplate. Match templates and return the result. 1:N matching are available.

public bool MatchTemplateMT(parameters)

	parameter	Description
1:N	IntPtr[] templateDB, int tcount, bool coreDetect, bool qulityCheck, out int index	Match multiple templates with template to be scanned. Return array index of templates which is matched with reference variable index.
	IntPtr[] templateDB, int tcount, byte[] currenttemplate, out int index	Match multiple templates and created template. Return array index of templates which is matched with reference variable index.

Parameters

template

Original template to be matched

templateDB

An array of original template to be matched. This sets starting address of each template as factor value.

coreDdetect

Detect the core of the fingerprint. In case of True, if the core of the fingerprint is not detected, an error value is returned.

qulityCheck

Check image quality. In case of True, if obtained image quality is poorer than ImageCheckQuality set in the Environment Setting stage, scan fails.

Return value

Success returns True, and failure returns False.

	ReturnCode	ReturnMessage
Success	1	Success Match
failure	0	Match Fail

		Matching template is NULL Template to be matched is NULL TimeOut

Example

MatchTemplateDB

Match template with template inserted in built-in DB and return user information in DataRow type. If data is not found, DataRow of null value is returned.

Compare template to be scanned with built-in DB and return matched information of the user. If scan is failed, NULL value is returned.
public DataRow MatchTemplateDB(bool coreDetect, bool qulityCheck)

Match temporary template with DB.
public DataRow MatchTemplateDB(byte[] template)

Parameters

Template
Template is 384-byte array to be matched
coreDdetect
Detect the core of the fingerprint. In case of True, if the core of the fingerprint is not detected, an error value is returned.
qulityCheck
Check image quality. In case of True, if obtained image quality is poorer than ImageCheckQuality set in the Environment Setting stage, scan fails

Return value

Success returns True, and failure returns False.

	ReturnCode	ReturnMessage
Success	1	Success Match
Failure	0	Image is not good Core is not detected in the center of the image Scan Fail Please Check your Sensor Put on your Finger Match Fail Matching template is NULL Table cannot be found.

		Fail to connect to DB.

Example

C#

```
//Create template to be compared.
byte[] temptemplate1 = new byte[384];
temptemplate1 = sfr.ScanTemplate(true, true);

DataRow dr = sfr.MatchTemplateDB(temptemplate1);

if (sfr.ReturnCode == 1)
{
        txtUserId.Text = dr[0].ToString();
        txtIndex.Text = dr[1].ToString();
}
```

VB.NET

```
//Create template to be compared.
Dim temptemplate1 (384) As Byte
temptemplate1 = sfr.ScanTemplate(true, true)

Dim dr As DataRow
dr = sfr.MatchTemplateDB(temptemplate1)

If sfr.ReturnCode = 1 Then
     txtUserId.Text = dr(0).ToString()
     txtIndex.Text = dr(1).ToString()
End If
```

MatchTemplateDBMT

MatchTemplateDBMT method used by users with Dual CPU, has the same method as MatchTemplateDB. Match template with template inserted in built-in DB and return user information in DataRow type. If data is not found, DataRow of null value is returned.

Compare template to be scanned with built-in DB and return matched information of the user. If scan is failed, NULL value is returned.

public DataRow MatchTemplateDBMT(bool coreDetect, bool qulityCheck)

Match temporary template with DB.

public DataRow MatchTemplateDBMT(byte[] template)

Parameters

Template

Template is 384-byte array to be matched

coreDdetect

Detect the core of the fingerprint. In case of True, if the core of the fingerprint is not detected, an error value is returned.

qulityCheck

Check image quality. In case of True, if obtained image quality is poorer than ImageCheckQuality set in the Environment Setting stage, scan fails

Return value

Success returns True, and failure returns False.

	ReturnCode	ReturnMessage
Success	1	Success Match
Failure	0	Image is not good Core is not detected in the center of the image Scan Fail Please Check your Sensor Put on your Finger Match Fail Matching template is NULL

		Table cannot be found. Fail to connect to DB.

Example

AbortMatch

Abort matching process. This method is used in multi-thread environment, and it can be used to abort matching when matching continues unlimitedly by setting Timeout property as 0.

public void AbortMatch()

Parameters

No parameters

Return value

No return value

	ReturnCode	ReturnMessage
Success	1	
Failure	0	

Example

4.5. Database

Explains the structure of built-in DB which uses Access (mdb) and how to select, insert, update and delete Data.

- CreateDatabase creates DB file of emroll.mdb in designated types (UserID field, fingerIndex field, memo field, 2 templates field).
- Search data values by using SselectTemplate method.
- Add user information and template data by using InsertTemple method.
- Update data excluding primary key by using UpdateTemplate method.
- Delete data by using DeleteTemplate method.
- ExportDB method is used when transferring data values inserted in built-in DB to another DB. Data file name is enroll.mdb and is created in a location designated by user.

CreateDataBase

This method creates DB to used built-in DB. It is a mandatory process when using built-in DB. File is created in a default folder where application program is run, and in case of ASP.NET, it is created in a location designated by programmer in WEB.CONFIG.

The name of DB: enroll.mdb
The name of Table: fingerprints
The name of Field

- UserID is a primary key of a table and less than 10-byte string type.
- FingerIndex is an index of an applicable finger. 10 fingers can be enrolled per UserID and its value in integer type can be designated by programmer. Number is assigned to each finger and it can be used by differentiating fingers. Finger number can't duplicate per UserID.
- Memo field is to be used as per necessity of programmer.
- Fingerprint1, Fingerprint2 are template data and Fingerprint1 is a required entry field.
- In case of not using TwoTemplate Mode, Null value of Fingerprint2 is saved.
- Table structure

Field name	Data type	Description	Declaration in application program
UserID	String/10byte	Primary Key	String
FingerIndex	Int	Finger Index	Int
Memo	String/50byte		String
Fingerprint1	OLE object	Template mandatory	Byte[] template1 = new byte[384]
Fingerprint2	OLE object	Template	Byte[] template2 = new byte[384]

public void CreateDataBase ()

Parameters

No parameters

Return value

Success returns True, and failure returns false.

	ReturnCode	ReturnMessage
Success	1	Create Success
failure	0	DB cannot be created.

Example

SelectTemplate

Return all Data of fingerprints table in data table type

public DataTable SelectTemplate()

Return applicable user's data in Data Table.

public DataTable SelectTemplate(string userId)

public DataTable SelectTemplate(string userId, int fingerIndex)

public DataTable SelectTemplate(string fromUserId, string toUserId)

Parameters

userId

User ID to search

fingerIndex

Fingerprint index of User ID

fromUserId

Start User ID to search

toUserId

End User ID to search

.

Return value

Return DataTable type

	ReturnCode	ReturnMessage
Success	1	If the number of data table row exceeds 0, return the number of row and if it's 0, table is empty..
failure	0	Check User ID Range Table can't be found. Connecting to DB has been failed

Example

Examples show that searched data values are connected to datagrid.

C#

```
DataTable datatable = sfr.SelectTemplate();
DataTable datatable = sfr.SelectTemplate(txtUserId.Text);
DataTable datatable = sfr.SelectTemplate(txtUserId.Text, txtTouser.Text);

dataGrid1.DataSource = datatable;
dataGrid1.Refresh();
```

VB.NET

```
Dim dTable As DataTable

dTable = sfr.SelectTemplate()
dTable = sfr.SelectTemplate(txtUserId.Text)
dTable = sfr.SelectTemplate(txtUserId.Text, txtTouser.Text)

dataGrid1.DataSource = dTable;
dataGrid1.Refresh();
```

InsertTemplate

Insert user information and template into DB. Template is created by using ScanTemplate method. As for user information field, refer to CreateDatabase method.

public bool InsertTemplate(string userid, int index, string memo, byte[] template1, byte[] template2)

Parameters

Userid

Mandatory user number in string type of less than 10 bytes

fingerIndex

Mandatory Integer-type fingerprint Index of UserID.

Memo

String type of no more than 50 bytes.

template1, template2

Template1 and template2 are byte array type and inserted as object type. Template2 can be Null but template1 is a mandatory value, so an error value is returned when Null value is transmitted.

Return value

Success returns True and failure returns False.

	ReturnCode	ReturnMessage
Success	1	Template Insert Success
Failure	0	Template Insert Fail userID is a mandatory field. template1 is mandatory. Exceeding maximum number(10) which applicable UserID can save. Table can't be found. Connecting to DB failed

Example

Code which saves user information and template. Null value of temptemplate2 is transferred.

C#

```
temptemplate1 = ScanTemplate(false, true);

bool rvalue = sfr.InsertTemplate(txtUserId.Text, Convert.ToInt32(txtIndex.Text),
txtMemo.Text, temptemplate1, temptemplate2);
```

VB.NET

```
Dim rvalue As Boolean

temptemplate1 = ScanTemplate(false, true)
rvalue = sfr.InsertTemplate(txtUserId.Text, Convert.ToInt32(txtIndex.Text), txtMemo.Text,
temptemplate1, temptemplate2);
```

DeleteTemplate

Delete data of applicable user number

public bool DeleteTemplate(string userid)
public bool DeleteTemplate(string userid, int fingerIndex)

Parameters

Userid
UserID to be deleted
fingerIndex
Finger Index to be deleted

Return value

Success returns True, and failure returns False.

	ReturnCode	ReturnMessage
Success	1	Template Delete Success
Failure	0	Cannot find UserId. Fail to connect to DB.

Example

UpdateTemplate

Update memo and template of applicable user number. UserID and fingerIndex, which are a primary key, cannot be updated.

public bool UpdateTemplate(string userid, int fingerindex, byte[] template1, byte[] template2)
public bool UpdateTemplate(string userid, int fingerindex, string memo)
public bool UpdateTemplate(string userid, int fingerindex, string memo, byte[] template, byte[] template2)

Parameters

userid
UserID to be updated
Fingerindex
Fingerindex to be updated.
Template1, Template2
Template data to be updated
memo
Memo to be updated

Return value

Success returns True, and failure returns False.

	ReturnCode	ReturnMessage
Success	1	Template Update Success
Failure	0	UserId can't be found. fingerIndex can't be found. Userid must be entered fingerIndex must be entered Template to be corrected is Null. Connecting to DB fails

Example

ExportDB

Transform all data in Fingerprints table into txt file format. Send location of file to be inserted to parameter and if there’s no folder, create a folder and save it.
File name is FigerPrints.txt.

public void ExportDB()

Parameters

No parameters

Return value

Success returns True, and failure returns False..

	ReturnCode	ReturnMessage
Success	1	Create Success
failure	0	Fail to create a file

Example

4.6. Image

Below methods enables users to read and insert the image and to change the size of the image.

- AdjustImage is a method to change width and height of obtained image.
- SaveImageFile is a method to insert the image to the location at the user’s disposal. Workable file formats include BMP, GIF, and JPG.
- ReadFileToBitmap is a method to read the inserted image file and to return it in bitmap type. It can read files in BMP, GIF and JPG format.

AdjustImage

Adjust the size of the image and transmit in Bitmap type.

public Bitmap AdjustImage(Bitmap bitmap, int picw, int pich)

Parameters

Bitmap

Bitmap of image to be adjusted

picw

width of image. Unit is pixel.

Pich

height of an image. Unit is pixel.

Return value

Success returns adjusted Bitmap, and failure returns null Bitmap.

	ReturnCode	ReturnMessage
Success	1	Adjust Success
failure	0	Adjust Fail.

Example

SaveImageFile

Save an image under name and in file format designated by user to a designated location. Optional formats include bmp, gif, jpg which can be found in ImageType data type.

public bool SaveImageFile(Image image, string fileName, int imageType)

Parameters

Image

Image to be saved

Filename

File name to be saved.

Imagetype

File format to be saved. Choose one from bmp, gif, jpg.

Return value

Success returns True, and failure returns False..

	ReturnCode	ReturnMessage
Success	1	Adjust Success
Failure	0	Wromg Image File Format You Must Input Directory You Must Input File Name

Example

Examples show how a user saves designated location, file format and file name through SaveFileDialog.

C#

```
        SaveFileDialog sFDialog = new SaveFileDialog();
        DialogResult result;
        string filename = "";
```

```
sFDialog.Filter = "Image File (*.bmp) | *.bmp |gif File (*.gif)|*.gif |jpg File (*.jpg)|*.jpg";
result = sFDialog.ShowDialog();

if (result == DialogResult.OK)
{
        filename = sFDialog.FileName;
        if (filename.Length > 0 )
        {
            sfr.SaveImageFile(pic5.Image, filename, sFDialog.FilterIndex);
        }
        else
            MessageBox.Show("Must Input save File name.");
}
```

VB.NET

```
Dim sFDialog As SaveFileDialog = New SaveFileDialog
Dim result As DialogResult
Dim filename As String = ""

sFDialog.Filter = "Image File (*.bmp) | *.bmp |gif File (*.gif)|*.gif |jpg File (*.jpg)|*.jpg"
result = sFDialog.ShowDialog()

If result = DialogResult.OK Then
    filename = sFDialog.FileName
    If filename.Length > 0 Then
        sfr.SaveImageFile(pic5.Image, filename, sFDialog.FilterIndex)
        WriteStatusBar()
    Else
        MessageBox.Show("Input save File name.")
    End If
End If
```

ReadFileToBitmap

Read image file and convert into bitmap.

public Bitmap ReadFileToBitmap(string filepath, int width, int height)

Parameters

Filepath

definite path of a file.

Width

width of bitmap

Height

height of bitmap

Return value

Success returns Bitmap, and failure returns Null Bitmap.

	ReturnCode	ReturnMessage
Success	1	Adjust Success
failure	0	You Must Input Directory You Must Input File Name File cannot be found

Example

Read a file saved in designated location and bring it in Bitmap type.

C#

```
OpenFileDialog oFDialog = new OpenFileDialog();
DialogResult result;
string filename = "";

oFDialog.Filter = "Image File (*.bmp) | *.bmp |gif File (*.gif)|*.gif |jpg File (*.jpg)|*.jpg";

result = oFDialog.ShowDialog();
if (result == DialogResult.OK)
```

```
{
    filename = oFDialog.FileName;
    if (filename.Length > 0 )
    {
        pic6.Image = sfr.ReadFileToBitmap(filename, pic3.Width, pic3.Height);
    }
   else
        MessageBox.Show("Choose Oprn File.");
}
```

VB.NET

```
Dim oFDialog As OpenFileDialog = New OpenFileDialog
Dim result As DialogResult
Dim filename As String = ""

oFDialog.Filter = "Image File (*.bmp) | *.bmp |gif File (*.gif)|*.gif |jpg File (*.jpg)|*.jpg"
result = oFDialog.ShowDialog()

If result = DialogResult.OK Then
    filename = oFDialog.FileName
    If filename.Length > 0 Then
        pic6.Image = sfr.ReadFileToBitmap(filename, pic3.Width, pic3.Height)
        WriteStatusBar()
    Else
        MessageBox.Show("Input save File name.")
    End If
End If
```

4.7. Property

There are total 23 properties including those of SFR class for sensor (10), scan & match (11) and message (2).

- Sensor related properties include getting a sensor currently in use or setting a sensor for use, setting sensitivity of a sensor and brightness of fingerprint image, etc.
- Scan & match related properties include getting quality of created fingerprint image, setting standby time for identification and standard image quality for quality check.
- Message related properties include ReturnCode and ReturnMessage properties which make available more detailed results of error and execution as SFR class delivers the result of method execution in code and character string after one method.

Device Property

Name	Description	Remark
Sensor	Set whether USB fingerprint reader is SFR300 ver.2, SFR300 or SFR200. Set when using SFR300 ver.2, SFR200-S and SFR300-S	SFRNONE, SFR200, SFR300 SFR300V2
Device	Get sensor index	Read/Write
DeviceNumber	Get the number of sensors	ReadOnly
DeviceSerial	Get the number of sensors	ReadOnly
SensorBrightness	Set or get sensitivity of sensor. The greater the number gets, the darker the sensor becomes. When SFR300 ver.2 or SFR300 is connected, default is 100. When SFR200 is connected, default is 50.	0 ~ 255
MaxSensorBrightness	Maximum brightness of sensor (255)	ReadOnly
MinSensorBrightness	Minimum brightness of sensor (0)	ReadOnly
SensorSensitivity	The higher the sensitivity gets, the more easily fingerprints can be accepted. The lower the sensitivity gets, the stabler fingerprint image can be obtained. Default is 4.	0 ~ 7
MaxSensorSensitivity	Maximum sensitivity of sensor (7)	ReadOnly
MinSensorSensitivity	Minimum sensitivity of sensor (0)	ReadOnly

Examplae

C#

```
// add sensors to combobox
cboSensor.Items.Add( Suprema.SFR.SensorType.SFRNONE.ToString());
cboSensor.Items.Add( Suprema.SFR.SensorType.SFR200.ToString());
cboSensor.Items.Add( Suprema.SFR.SensorType.SFR300.ToString());
cboSensor.Items.Add( Suprema.SFR.SensorType.SFR300V2.ToString());
cboSensor.SelectedIndex = 3;
```

```
// Set sensor selected from combobox as sfr object sensor.
sfr.Sensor = cboSensor.SelectedItem.ToString();

// add fingerprint brightness into combobox
for (int i = sfr.MinSensorBrightness; i <= sfr.MaxSensorBrightness; i += 5)
          cboB.Items.Add(i);
cboB.SelectedIndex = 20;

sfr.SensorBrightness = Convert.ToInt32(txtBriteness.Text);

// Add sensor sensitivity available for setting to combo box
for (int i = sfr.MinSensorSensitivity; i <= sfr.MaxSensorSensitivity; i++)
          cboS.Items.Add(i);
cboS.SelectedIndex = 4;
sfr.SensorSensitivity = Convert.ToInt32(txtSensitivity.Text);
```

Scan & Match Property

Name	description	Remark
Quality	Visibility factor of template image. After call of ScanTemplate, image visibility property can be used.	ReadOnly
MatchMode	Set or get Match mode. Default value is Nomal_mode. Selecting Fast_Mode enables the identification process to speed up.	Nomal_Mode, Fast_Mode, Invalid_Mode
TimeOut	Set or get stand-by time during matching. Setting it 0 makes it possible to proceed identification process without time limit.	0 ~ 10
MaxTimeOut	Maximum stand-by time (10)	ReadOnly
MinTimeOut	Minimum stand-by time (0)	ReadOnly
SecurityLevel	Set or get identification level at a time of identification. Default value is 4. False Accept Rate(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) Level 4 indicates that one out of 100000 attempts to enroll fingerprint can be an error.	1 ~ 7
MaxSecurityLevel	Maximum identification degree (7)	ReadOnly
MinSecurityLevel	Minimum identification degree (1)	ReadOnly
ImageCheckQuality	Set or get standard visibility factor of image. Default is 70. In case of True,	30 ~ 100%

	compared value between Scantemplate and qualityCheck	
MaxImageCheckQuality	Maximum image visibility (100%)	ReadOnly
MinImageCheckQuality	Minimum image visibility (30%)	ReadOnly

Example

Set property with values selected from Combobox and check the property values.

C#

```
for (int i = sfr.MinTimeOut; i <= sfr.MaxTimeOut; i++)
        cboT.Items.Add(i);
cboT.SelectedIndex = 5;
sfr.TimeOut = Convert.ToInt32(cboT.SelectedItem.ToString());
txtTimeout.Text    = sfr.TimeOut.ToString();

for (int i = sfr.MinSecurityLevel; i <= sfr.MaxSecurityLevel; i++)
        cboL.Items.Add(i);
cboL.SelectedIndex = 4;
sfr.SecurityLevel = Convert.ToInt32(cboL.SelectedItem.ToString());
txtSlevel.Text = sfr.SecurityLevel.ToString();

for (int i = sfr.MinImageCheckQuality; i <= sfr.MaxImageCheckQuality; i += 10)
        cboQ.Items.Add(i);
cboQ.SelectedIndex = 1;
sfr.ImageCheckQuality = Convert.ToInt32(cboQ.SelectedItem.ToString());
txtQuality.Text      = sfr.ImageCheckQuality.ToString();

cboMatch.Items.Add( Suprema.SFR.MatchType.NORMAL_MODE.ToString());
cboMatch.Items.Add( Suprema.SFR.MatchType.FAST_MODE.ToString());
cboMatch.Items.Add( Suprema.SFR.MatchType.INVALID_MODE.ToString());
cboMatch.SelectedIndex =0;
```

Message Property

Name	Description	Remark
ReturnCode	Code of implementing method after the final method is carried out. Refer to Appendix A.	ReadOnly
ReturnMessage	Message of implementing method after the final method is carried out.	ReadOnly

Example

The following shows message values indicated in status bar.

C#

```
this.statusBarPanel1.Text = "Message : " + sfr.ReturnMessage;
this.statusBarPanel2.Text = "Code : "       + sfr.ReturnCode.ToString();
```

4.8. AssemblyInfo

AssemblyInfo

Bring information on SDK together.

public string[] GetAssemblyInfo()

Parameters

No parameters

Return value

String array	Description	Remark
0	Name	UniFinger Engine for .NET SDK
1	Version	Version 2.6.0.0
2	Date of Creation	Copyright 2006 Suprema Inc. All rights reserved.
3	Description of SDK	UniFinger Engine for .NET SDK 2.6

Example

C#

```
string[] info = sfr.GetAssemblyInfo();
txtVersion.Text = info[0].ToString() + " " + info[1].ToString();
txtCo.Text        = info[2].ToString();
```

VB.NET

```
Dim info() As String
info = sfr.GetAssemblyInfo()
txtVersion.Text = info(0).ToString() + " " + info(1).ToString()
txtCo.Text = info(2).ToString()
```

Appendix A Constant List

Message shown according to returned value.

Code Value	Description	Remark
1002	Initialize Fail	
1004	Please Check your Sensor	
1005	Put on your Finger	
1007	Scan Fail	
1009	Select Reader Fail	
1011	Rotate Fail	
1021	Two template mode. Once more put on your finger	
1101	Check Brightness Value (0 ~ 255)	
1102	Check Sensitivity Value (0 ~ 7)	
1103	Check Timeout Value (0 ~ 10 second)	
1104	Check Security Level (1 ~ 7)	
1105	Check Quality Value (30 ~ 100)	
1202	Not Match Finger	
1203	Please Check your source finger	
1204	Please Check your current Finger	
1301	Image Save Success	
1302	Wrong Image File Format	
1303	You Must Input File Name	
1305	Extract Template Fail	
1037	Adjust Fail	
1501	You Must Input User ID	
1502	You Must Input Finger Template	
1602	Template Insert Fail	
1603	Template Is Empty	
1604	You Must Input User ID	

1605	Check User ID Range	
1609	Not Found user ID	
1610	Not Found user ID Or Finger Index	
1611	Not open Database	
1612	Database is Empty	
1613	Fingerprints Table is Empty	
1614	No Data Found	
1651	Create Database Success	
1652	Database exits	