

eNBSP - NBioBSP

NITGEN&COMPANY Biometric Service Provider SDK

Programmer's Manual .NET

SDK version 5.0x

© Copyright 2000-2013 NITGEN&COMPANY Co., Ltd.

ALL RIGHTS RESERVED

Serial Number:

Specifications subject to change without notice.

“NITGEN”, the NITGEN logo, “eNBSP”, “NBioBSP”, “NBioAPI”, “NITGEN Fingkey Mouse”, “eNDeSS”, “eNFolder”, and “eNFile” are trademarks of NITGEN&COMPANY Co., Ltd. All other brands or products may be trademarks or service marks of their respective owners.

INDEX

CHAPTER 1. INTRODUCTION.....	5
1.1 SUPPORT MODULES.....	5
1.2 SAMPLE PROGRAMS.....	6
1.2.1 <i>Support samples (32bit SDK)</i>	6
1.2.2 <i>Support Samples (64bit SDK)</i>	6
CHAPTER 2. .NET PROGRAMMING	7
2.1 MODULE INITIALIZATION AND CLOSURE.....	7
2.1.1 <i>Module initialization</i>	7
2.1.2 <i>Module closure</i>	7
2.2 DEVICE RELATED PROGRAMMING	8
2.2.1 <i>Listing devices</i>	8
2.2.2 <i>Initializing the device</i>	9
2.2.3 <i>Closing the device</i>	9
2.3 FINGERPRINT ENROLLMENT	10
2.4 FINGERPRINT VERIFICATION	11
2.5 CLIENT / SERVER ENVIRONMENT PROGRAMMING.....	12
2.5.1 <i>Fingerprint enrollment</i>	12
2.5.2 <i>Fingerprint verification</i>	13
2.6 USING PAYLOAD	14
2.6.1 <i>Inserting payload in fingerprint data</i>	14
2.6.2 <i>Extracting payload from fingerprint Template</i>	15
2.7 CHANGING THE NBIOSPP USER INTERFACE.....	16
2.6 HOW TO GET NFIQ INFORMATION	16
2.6.1 <i>How to get NFIQ information using AUDIT (Fingerprint Image)</i>	16
2.6.2 <i>How to get NFIQ information using RAW data</i>	16
2.6.3 <i>How to get NFIQ information using FIR handle</i>	17
2.6.4 <i>How to use QUALITY_INFO_0 Class</i>	17
2.7 IMAGE FORMAT CONVERSION	18
2.8 ISO 19794-4 FORMAT CONVERSION.....	19
APPENDIX A. CLASS LIBRARY FOR .NET REFERENCE	20
A.1 NBIOAPI CLASS.....	20
A.1.1 <i>Basic Method</i>	20
A.1.2 <i>Memory Method</i>	25
A.1.3 <i>BSP Method</i>	28
A.1.4 <i>User Interface Method</i>	40
A.1.5 <i>NFIQ Method</i>	40
A.1.6 <i>Image Convert Method</i>	41

A.1.7 ISO 19794-4 Method/Class/Struct.....	43
A.2 NBIOAPI.EXPORT CLASS	46
A.3 NBIOAPI.INDEXSEARCH CLASS	51
A.3.1 Initialization Method.....	51
A.3.2 Enroll / Remove / Search Method.....	53
A.3.2 DB Method.....	56
A.4 NBIOAPI.NSEARCH CLASS.....	58
A.4.1 Initialization Method.....	58
A.4.2 Enroll / Remove / Search Method.....	60
A.4.2 DB Method.....	64

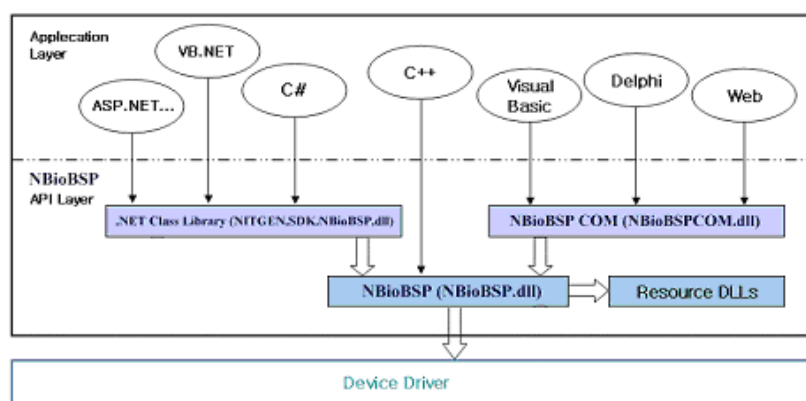
Chapter 1. Introduction

The eNBSP (NBioBSP) SDK provides feature rich, high-level functionality that can be integrated into any application requiring fingerprint authentication. NBioBSP technology is built on the NBioAPI™ specification, working seamlessly with the most durable, compact, and reliable optics-based fingerprint readers in the world.

All NBioBSP SDK components contain the APIs needed for biometric authentication of multiple users and device functions. NBioBSP is equipped with self-contained User Interfaces for enrollment and verification, enabling software application developers to quickly and easily integrate fingerprint authentication into the application of their choice.

1.1 Support Modules

- **NBioBSP.dll**
This is the main module of the NBioBSP that implements all of NITGEN's biometric functions including fingerprint enrollment and verification
- **NITGEN.SDK.NBioBSP.dll**
The NBioBSP Class Library (NITGEN.SDK.NBioBSP.dll) is designed to support developers using C#, VB.NET, ASP.NET, J# and the like in Microsoft .NET environment.
The NBioBSP Class Library also uses NBioBSP.dll and provides higher level of interfaces. NBioBSP Class Library supports almost all NBioBSP functions.



[Development model using NBioBSP SDK]

1.2 Sample programs

1.2.1 Support samples (32bit SDK)

- C#
 - BSPDemoCS – Basic function demo application for C#.
 - UITestCS – User Interface demo application for C#.
 - IndexSerchDemoCS – IndexSearch demo application for C#.
 - RollDemoCS – Roll fingerprint function demo application for C#.

1.2.2 Support Samples (64bit SDK)

- C#
 - BSPDemoCS – Basic function demo application for C#.
 - UITestCS – User Interface demo application for C#.
 - IndexSerchDemoCS – IndexSearch demo application for C#.
 - BSPRollDemoCS – Roll fingerprint function demo application for C#.
 - ExportDemoCS – Data Export / Import function demo application for C#.
- VB.NET
 - BSPDemoVBNET – Basic function demo application for VB.NET.
 - IndexSerchDemoVBNET – IndexSearch demo application for VB.NET.

Chapter 2. .NET Programming

This chapter describes how to make a .NET programming with the NBioBSP Class Library that is, named "NITGEN.SDK.NBioBSP.dll", designed and developed to support Microsoft .NET environment on C#, VB.NET, ASP.NET, J# and the like. The NBioBSP Class Library also uses NBioBSP.dll and provides higher level of interfaces. NBioBSP Class Library supports almost all NBioBSP functions.

While a number of languages support the .NET programming, this section will introduce the usage of C# programming that is the most popular of .NET languages. It is also applicable to other .NET languages because most parts have not much difference besides initialization.

2.1 Module initialization and closure

2.1.1 Module initialization

Use the following code to initialize the NBioBSP Class Library module.

```
using NITGEN.SDK.NBioBSP;
...
m_NBioAPI = new NBioAPI();
...
```

2.1.2 Module closure

No specific code is needed to close or free any memory on .NET language.(During a collection, the garbage collector will free.)

In scenarios where resources must be released at a specific time, developer can call dispose method.

```
...
m_NBioAPI.Dispose();
...
```

Use this method to release unmanaged resources such as handles and memories held by an instance of the NBioBSP object.

2.2 Device related programming

The device must be opened before it can be used.

Use the **Enumerate** method to determine which device is linked to the system.

2.2.1 Listing devices

Before opening the device, use the **EnumerateDevice** method to determine the number and type of devices linked to the PC. Once this is activated, the number of devices linked to the PC and the ID for each device will be returned.

DeviceID is composed of the device names and their instance numbers.

DeviceID = Instance Number + Device Name

If there is only one device for each type in the system, the instance number will be '0.' In this way, the device name has the same value as the device ID.

The following is an example of how to use the **EnumerateDevice** method. All devices found by this method will be added into the combo box, `comboDevice`.

```
m_NBioAPI = new NBioAPI();
...
int i;
uint nNumDevice;
short[] nDeviceID;
NBioAPI.Type.DEVICE_INFO_EX[] deviceInfoEx;

uint ret = m_NBioAPI.EnumerateDevice(out nNumDevice, out nDeviceID, out deviceInfoEx);

if (ret == m_NBioAPI.Error.NONE)
{
    comboDevice.Items.Add("Auto_Detect");
    for (i = 0; i < nNumDevice; i++)
    {
        comboDevice.Items.Add(deviceInfoEx[i].Name);
    }
}
```

The device ID will be returned if the number of devices is entered in the DeviceNumber property of the `nDeviceID` (DeviceNumber). For example, `nDeviceID(0)` will show the DeviceID of the first device.

2.2.2 Initializing the device

The **OpenDevice** method is used to initialize the device for the NBioBSP Class Library. Device initialization must be done using the **OpenDevice** method before device related functions such as enrolling, verifying, and capturing will work properly.

In the event that you are unsure which devices have been installed, use the **EnumerateDevice** method to determine what devices have previously been installed.

```
m_NBioAPI = new NBioAPI();
...
ret = m_NBioAPI.OpenDevice(DeviceID);

if (ret == NBioAPI.Error.NONE)
    // Open device success ...
else
    // Open device failed ...
```

The device can be set automatically using **NBioAPI.Type.DEVICE_ID.AUTO**.

This setting will search the latest active device if there are multiple devices connected.

```
m_NBioAPI.OpenDevice(NBioAPI.Type.DEVICE_ID.AUTO);
```

NBioBSP_DEVICE_ID_AUTO_DETECT use the latest opened device.

2.2.3 Closing the device

The **CloseDevice** method should be used to close the device. The same **DeviceID** used to call the **Open** method must be used again to call the **CloseDevice** method.

```
m_NBioAPI = new NBioAPI();
...
ret = m_NBioAPI.CloseDevice(DeviceID);

if (ret == NBioAPI.Error.NONE)
    // Close device success ...
else
    // Close device failed ...
```

The current device must be closed before opening another device.

2.3 Fingerprint enrollment

The **Enroll** method is used to enroll fingerprints. All fingerprint data is used as the type of handle, binary or encoded text found in the **NBioBSP Class Library** module.

Fingerprint data will be entered into the handle of **FIR** property upon successful enrollment, and it can be returned as the type of binary or encoded text.

The NBioBSP Class Library provides various overloading **Enroll** method that can be used for the specific purpose. One of example is as below.

```
m_NBioAPI = new NBioAPI();
...
NBioAPI.Type.HFIR hNewFIR;

ret = m_NBioAPI.Enroll(out hNewFIR, null);

if (ret == NBioAPI.Error.NONE)
{
    // Enroll success ...

    // Get binary encoded FIR data
    NBioAPI.Type.FIR biFIR;
    m_NBioAPI.GetFIRFromHandle(hNewFIR, out biFIR);

    // Get text encoded FIR data
    NBioAPI.Type.FIR_TEXTENCODING textFIR;
    m_NBioAPI.GetTextFIRFromHandle(hNewFIR, out textFIR, true);

    // Write FIR data to file or DB
}
else
    // Enroll failed ...
```

Fingerprint data will be stored as saving **biFIR** or **textFIR** to a file or DB.

2.4 Fingerprint verification

The **Verify** method performs fingerprint verification using the existing fingerprint data as a comparison with newly input fingerprints, and returns the verification result. After successful verification, the method returns the payload if it is available.

```
m_NBioAPI = new NBioAPI();
...
//Read FIRText Data from File or DB.
...

uint ret;
bool result;
NBioAPI.Type.FIR_PAYLOAD myPayload = new NBioAPI.Type.FIR_PAYLOAD();

// Verify with binary FIR
ret = m_NBioAPI.Verify(biFIR, out result, myPayload);

if (ret != NBioAPI.Error.NONE)
{
    // Verify Success

    // Check payload
    if (myPayload.Data != null)
    {
        textPayload.Text = myPayload.Data;
    }
}
else
    // Verify failed
```

2.5 Client / Server environment programming

Unlike standalone environments, the fingerprint enrollment and matching occur in separate places within the Client/Server environment. Fingerprints are generally enrolled in the client and later matched in the Server.

The **Enroll** method registers fingerprints while the **Capture** method verifies fingerprints.

The **VerifyMatch** method matches fingerprints in the Server through the use of previously registered fingerprints from the client.

2.5.1 Fingerprint enrollment

Use the **Enroll** method for fingerprint enrollment in the client.

```
m_NBioAPI = new NBioAPI();
...
NBioAPI.Type.HFIR hNewFIR;

ret = m_NBioAPI.Enroll(out hNewFIR, null);

if (ret == NBioAPI.Error.NONE)
{
    // Enroll success ...

    // Get binary encoded FIR data
    NBioAPI.Type.FIR biFIR;
    m_NBioAPI.GetFIRFromHandle(hNewFIR, out biFIR);

    // Get text encoded FIR data
    NBioAPI.Type.FIR_TEXTENCODE textFIR;
    m_NBioAPI.GetTextFIRFromHandle(hNewFIR, out textFIR, true);

    // Write FIR data to file or DB
}
else
    // Enroll failed ...
```

2.5.2 Fingerprint verification

Use the **Capture** method for registering only one fingerprint in the client. While the **Enroll** method allows several fingerprints to be enrolled and transferred in the FIR, the **Capture** method registers only one fingerprint.

The **Capture** method takes the purpose of fingerprint capture and allows NBioAPI.Type.FIR_PURPOSE.VERIFY only as a parameter.

```
m_NBioAPI = new NBioAPI();
...

ret = m_NBioAPI.Capture(NBioAPI.Type.FIR_PURPOSE.VERIFY, out hCapturedFIR,
                        NBioAPI.Type.TIMEOUT.DEFAULT, null, null);

if (ret == NBioAPI.Error.NONE)
    // Capture success ...
else
    //Capture failed ...
```

The **VerifyMatch** method matches fingerprints stored on the Server.

The **VerifyMatch** method takes two parameters, FIR received from client and FIR stored in server. After successful verification, the method returns the payload.

The payload comes from the second parameter, **StoredFIR**, and does not affect to the payload of the first parameter, **CapturedFIR**.

```
m_NBioAPI = new NBioAPI();
...
// Get Captured FIR Data from Client and Read stored FIR Data from File or DB.
...

uint ret;
bool result;
NBioAPI.Type.FIR_PAYLOAD myPayload = new NBioAPI.Type.FIR_PAYLOAD();

ret = m_NBioAPI.VerifyMatch(hCapturedFIR, hStoredFIR, out result, myPayload);

if (ret != NBioAPI.Error.NONE)
{
    // Verify Success

    // Check payload
    if (myPayload.Data != null)
    {
        textPayload.Text = myPayload.Data;
    }
}
else
    // Verify failed
```

2.6 Using Payload

Including other data within the fingerprint data is called a **Payload**.

2.6.1 Inserting payload in fingerprint data

At the time of fingerprint enrollment, use the **Enroll** method to include payload with the FIR. The **CreateTemplate** method can be used to insert payload into an existing FIR.

The **Enroll** method will use the fingerprint data and payload to provide a parameter for later comparison.

```
m_NBioAPI = new NBioAPI();
...
NBioAPI.Type.HFIR hNewFIR;
NBioAPI.Type.FIR_PAYLOAD myPayload = new NBioAPI.Type.FIR_PAYLOAD();
myPayload.Data = "Your Payload Data";

ret = m_NBioAPI.Enroll(out hNewFIR, myPayload);

if (ret == NBioAPI.Error.NONE)
    // Enroll success ...
else
    // Enroll failed ...
```

Use the **CreateTemplate** method to insert a **payload** into existing fingerprint data.

The **CreateTemplate** method can also add new fingerprint data onto existing fingerprint data.

```
m_NBioAPI = new NBioAPI();
...
NBioAPI.Type.HFIR hNewFIR;
NBioAPI.Type.FIR_PAYLOAD myPayload = new NBioAPI.Type.FIR_PAYLOAD();
myPayload.Data = "Your Payload Data";

ret = m_NBioAPI.CreateTemplate(null, hStoredFIR, out hNewFIR, myPayload);

if (ret == NBioAPI.Error.NONE)
    // CreateTemplate success ...
else
    // CreateTemplate failed ...
```

2.6.2 Extracting payload from fingerprint Template

Payload in fingerprint templates (registered data) will only be extracted if matched using the **Verify** method or if the **VerifyMatch** method is true.

```
m_NBioAPI = new NBioAPI();
...
//Read FIRText Data from File or DB.
...

uint ret;
bool result;
NBioAPI.Type.FIR_PAYLOAD myPayload = new NBioAPI.Type.FIR_PAYLOAD();

// Verify with binary FIR
ret = m_NBioAPI.Verify(biFIR, out result, myPayload);

if (ret != NBioAPI.Error.NONE)
{
    // Verify Success

    // Check payload
    if (myPayload.Data != null)
    {
        textPayload.Text = myPayload.Data;
    }
}
else
    // Verify failed
```

Extracting payloads using the **VerifyMatch** method is the same as using the **Verify** Method.

2.7 Changing the NBioBSP User Interface

The **NBioBSP Class Library** module offers resource files for customization of the basic UI in English.

Use the **SetSkinResource** method to load UI resources in languages other than English.

```
m_NBioAPI = new NBioAPI();
...
string szSkinFileName;
openFileDialog.Filter = "DLL files (*.dll)|*.dll|All files (*.*)|*.*";

if (openFileDialog.ShowDialog(this) == DialogResult.OK)
{
    szSkinFileName = openFileDialog.FileName;

    if (szSkinFileName.Length != 0)
    {
        // Set skin resource
        bool bRet = m_NBioAPI.SetSkinResource(szSkinFileName);

        if (bRet)
            labStatus.Text = "Set skin resource Success!";
        else
            labStatus.Text = "Set skin resource failed!";
    }
}
```

Resource files must have an absolute path. Extra documents are offered for making customized UI's.

2.6 How to get NFIQ information

Provides with three methods to support NIST Fingerprint Image Quality data.

- Method for obtaining NFIQ data using AUDIT information
- Method for obtaining NFIQ data using RAW data
- Method for obtaining NFIQ data using FIR handle

NFIQ value ranges from 1(lowest quality) to 5(highest quality)

2.6.1 How to get NFIQ information using AUDIT (Fingerprint Image)

If you could have gotten `NBioAPI.Export.EXPORT_AUDIT_DATA` using `Export.NBioBSPToImage` method, you can also get NFIQ information using `NBioAPI.Type.QUALITY_INFO_0` Class type output parameter of `NBioAPI.GetNFIQInfo` method.

```
NBioExport.NBioBSPToImage(m_FIRCaptureAudit, out exportAuditData);

NBioAPI.Type.QUALITY_INFO_0 QualityInfo;
ret = m_NBioAPI.GetNFIQInfo(exportAuditData, out QualityInfo);
```

2.6.2 How to get NFIQ information using RAW data

To get NFIQ information using one fingerprint image, you can use `NBioAPI.GetNFIQInfoFromRaw` method.


```
int nNFIQ;  
m_NBioAPI.GetNFIQInfoFromRaw(rawData  
    , imageWidth  
    , imageHeight  
    , out nNFIQ);
```

2.6.3 How to get NFIQ information using FIR handle

To get NFIQ information using FIR handle, not image information, you can use NBioAPI.Type.QUALITY_INFO_0 Class type output parameter of the NBioAPI.GetNFIQInfoFromHandle method.

```
NBioAPI.Type.HFIR m_FIRCaptureAudit;  
m_NBioAPI.Capture(NBioAPI.Type.FIR_PURPOSE.VERIFY, out m_FIRCapture, -1, m_FIRCaptureAudit, null);  
  
NBioAPI.Type.QUALITY_INFO_0 QualityInfo;  
m_NBioAPI.GetNFIQInfoFromHandle(m_FIRCaptureAudit, out QualityInfo);
```

2.6.4 How to use QUALITY_INFO_0 Class

To get N number of fingerprint image information, you should use QUALITY_INFO_0 Class and the information is stored in a Quality attribute array. Quality array is two-dimensional array of 11 rows and 2 columns. The first one specifies Finger ID and the second one means SampleNum.

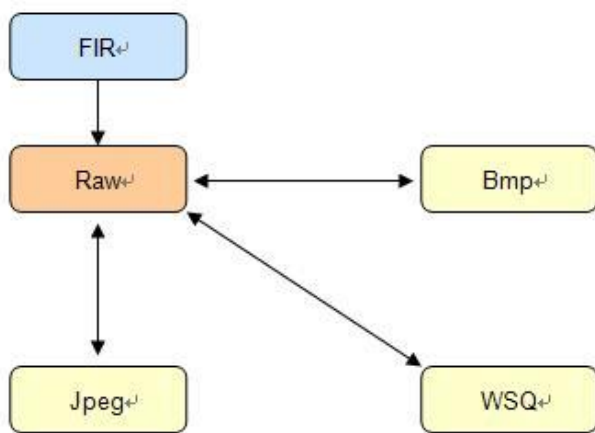
```
for (int f = 0; f < QualityInfo.Quality.GetLength(0); f++)  
{  
    for (int s = 0; s < QualityInfo.Quality.GetLength(1); s++)  
    {  
        if (QualityInfo.Quality[f, s] != NBioAPI.Type.QUALITY.QUALITY_NONE)  
            strResult = "Quality[" + f + "," + s + "]: " + QualityInfo.Quality[f, s];  
    }  
}
```

2.7 Image Format Conversion

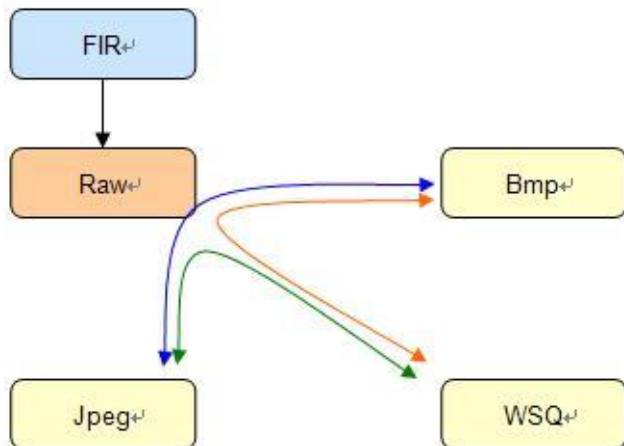
NBioAPI Class provides with methods for converting formats such as BMP, Jpeg, WSQ, including RAW image.

Format	Method
Raw Buffer → Bmp Buffer	ImgConvRawToBmpBuf
Bmp Buffer → Raw Buffer	ImgConvBmpToRawBuf
Raw Buffer → Jpeg Buffer	ImgConvRawToJpgBuf
Jpeg Buffer → Raw Buffer	ImgConvJpgToRawBuf
Raw Buffer → WSQ Buffer	ImgConvRawToWSQBuf
WSQ Buffer → Raw Buffer	ImgConvWSQToRawBuf

Each image can be converted from its initial format from FIR. Current Buffer is given on the basis that it is converted to Buffer. A few formats can be converted interchangeably from Raw image as follows.



To convert between other formats, not from/to its initial format, you can convert them on the basis of Raw image. For example, you can use `ImgConvBmpToRawBuf` and `ImgConvRawToJpgBuf` to convert BMP to Jpeg.



2.8 ISO 19794-4 Format Conversion

NBioAPI Class provides with methods for converting formats to support ISO 19794-4 data.

- Fingerprint image data → ISO Buffer: ExportRawToISOV1, ExportRawToISOV2
- ISO Buffer → Raw Buffer: ImportISOToRaw

The conversion process contains the following.

```
NBioAPI.Export.EXPORT_AUDIT_DATA exportAuditData;
NBioAPI.Export NBioExport = new NBioAPI.Export(m_NBioAPI);

String strResult;

NBioExport.NBioBSPToImage(m_FIREnrollAudit, out exportAuditData);

byte[] outBuf;
NBioAPI.NIMPORTRAWSET rawset;

outBuf = null;
NBioAPI.ExportRawToISOV1(exportAuditData, false, NBioAPI.COMPRESS_MOD.NONE, out outBuf);

NBioAPI.ImportISOToRaw(outBuf, out rawset);

strResult = "    - Rawset["+i+"] FingerID: "+ rawset.ImportRawData[i].nFingerID
            + ", ImgHeight:" + rawset.ImportRawData[i].nImgHeight
            + ", ImgWidth:" + rawset.ImportRawData[i].nImgWidth
            + ", RawLength:" + rawset.ImportRawData[i].RawData.Length;

for (int f = 0; f < exportAuditData.FingerNum; f++)
{
    for (int s = 0; s < exportAuditData.SamplesPerFinger; s++)
    {
        int fingerID = exportAuditData.AuditData[f].FingerID;
        uint imageWidth = exportAuditData.ImageWidth;
        uint imageHeight = exportAuditData.ImageHeight;

        outBuf = null;
        ret = NBioAPI.ExportRawToISOV2((byte)fingerID, (ushort)imageWidth, (ushort)imageHeight
            , exportAuditData.AuditData[f].Image[s].Data, false
            , NBioAPI.COMPRESS_MOD.NONE, out outBuf);
    }
}
```

Appendix A. Class Library for .NET Reference

A.1 NBioAPI Class

As a main class of the NBioBSP Class Library, this class includes many functions and some other classes. This object must be declared.

A.1.1 Basic Method

GetVersion

```
public System.UInt32 GetVersion ( out NITGEN.SDK.NBioBSP.NBioAPI.Type.VERSION Version );
```

Description

This function is to retrieve the current version of the BSP module.

Parameters

Version:

Version information.

Return Value

NBioAPI.Error.NONE : No error

GetInitInfo

```
public System.UInt32 GetInitInfo ( out NITGEN.SDK.NBioBSP.NBioAPI.Type.INIT_INFO_0 InitInfo );
```

Description

This function is to retrieve the initialization settings.

Parameters

InitInfo:

A object to a class that receives the initialization settings.

Return Value

NBioAPI.Error.NONE : No error.

SetInitInfo

```
public System.UInt32 SetInitInfo ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INIT_INFO_0 InitInfo );
```

Description

This function is to configure the initialization settings. The NBioAPI supports the type 0 structure. The structure, InitInfo, must be initialized before use.

Parameters

InitInfo:

A object to a class containing the initialization settings.

Return Value

NBioAPI.Error.NONE : No error.

NBioAPI.Error.INIT_MAXFINGER : Invalid value for the MaxFingersForEnroll.

NBioAPI.Error.INIT_SAMPLESPERFINGER : Invalid SampleNumber.

NBioAPI.Error.INIT_ENROLLQUALITY : Invalid EnrollQuality.

NBioAPI.Error.INIT_VERIFYQUALITY : Invalid VerifyQuality.

NBioAPI.Error.INIT_IDENTIFYQUALITY : Invalid Identify Quality.

NBioAPI.Error.INIT_SECURITYLEVEL : Invalid security level.

EnumerateDevice

```
public System.UInt32 EnumerateDevice (out System.UInt32 NumDevice , out short[] DeviceID );
```

Description

This function is to retrieve the number of devices and device IDs attached to the system.

Parameters

NumDevice:

The number of devices attached.

DeviceID:

An array including the device IDs.

Return Value

NBioAPI.Error.NONE : No error.

GetDeviceInfo

```
public System.UInt32 GetDeviceInfo ( System.Int16 DeviceID ,  
                                   out NITGEN.SDK.NBioBSP.NBioAPI.Type.DEVICE_INFO_0 DeviceInfo );
```

Description

This function retrieves information about the specified device.

Parameters

DeviceID:

The device ID to be queried.

DeviceInfo:

A object to a class that receives the device information.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.DEVICE_NOT_OPENED: Device not opened.

NBioAPI.Error.WRONG_DEVICE_ID: Invalid device ID.

SetDeviceInfo

```
public System.UInt32 SetDeviceInfo ( System.Int16 DeviceID ,  
                                    NITGEN.SDK.NBioBSP.NBioAPI.Type.DEVICE_INFO_0 DeviceInfo );
```

Description

This function is to configure the specific options of the device attached to the system.

The image width and height in the structure are read-only.

Parameters

DeviceID:

The device ID to be configured.

DeviceInfo:

A object to a class specifying the device information.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.DEVICE_NOT_OPENED: Device not opened.

NBioAPI.Error.WRONG_DEVICE_ID: Invalid device ID.

NBioAPI.Error.DEVICE_BRIGHTNESS: Invalid device brightness.

NBioAPI.Error.DEVICE_CONTRAST: Invalid device contrast.

NBioAPI.Error.DEVICE_GAIN: Invalid device gain.

OpenDevice

```
public System.UInt32 OpenDevice ( System.Int16 DeviceID )
```

Description

This function is to initialize the device. Entering a value of 0 for the DeviceID means that a default device will be used. It must be called to use the device related functions such as Capture or Enroll.

Parameters

nDeviceID:

The device ID to open. A 0 value means the default device will be used.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.DEVICE_ALREADY_OPENED: Device already opened.

NBioAPI.Error.DEVICE_OPEN_FAIL: Failed to open device.

CloseDevice

```
public System.UInt32 CloseDevice ( System.Int16 DeviceID );
```

Description

This function is to close the device opened by the OpenDevice function. The value of the parameter, DeviceID, must be identical to the one used for the OpenDevice function.

Parameters

DeviceID:

The device ID to be closed.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.DEVICE_NOT_OPENED: Device not opened.

NBioAPI.Error.WRONG_DEVICE_ID: Invalid device ID.

AdjustDevice

```
public System.UInt32 AdjustDevice ( NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );
```

Description

This function is used to configure the brightness of the device. It prompts a dialog on which users can change the brightness and contrast of the device.

Parameters

WindowOption:

A object to a class containing the window options of the NBioBSP module.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.DEVICE_NOT_OPENED: Device not opened.

NBioAPI.Error.WRONG_DEVICE_ID: Invalid device ID.

NBioAPI.Error.USER_CANCEL: Cancelled by user during adjustment.

GetOpenedDeviceID

```
public System.Int16 GetOpenedDeviceID ( );
```

Description

This function returns the device ID currently opened.

Parameters

N/A

Return Value

Returns the device ID currently opened.

CheckFinger

```
public System.UInt32 CheckFinger (System.Boolean bExistFinger );
```

Description

This function is to check if a finger is placed on the fingerprint sensor. Only valid for USB fingerprint devices.

Support: HFDU 01/04/06(device driver version 4.1.0.1 or higher), HFDU 08/09, HFDU 11/14

Not support: HFDU 05/07.

Parameters

bExistFinger:

A object to Boolean indicating whether a finger is placed on the fingerprint sensor or not. If yes, it returns NBioAPI_TRUE, otherwise NBioAPI_FALSE.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.DEVICE_NOT_OPENED: Device not opened.

NBioAPI.Error.LOWVERSION_DRIVER: Not supported for your device driver.

A.1.2 Memory Method

GetFIRFromHandle

```
public System.UInt32 GetFIRFromHandle ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR hFIR ,  
                                         out NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR FIR );  
  
public System.UInt32 GetFIRFromHandle ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR hFIR ,  
                                         out NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR FIR ,  
                                         NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_FORMAT Format );
```

Description

This function is to receive a FIR from the FIR handle. An application must allocate the buffer for the NBioAPI_FIR structure. Note that this function has two different types.

Parameters

hFIR:

The handle of the FIR.

FIR:

FIR class that receives fingerprint information.

Format:

FIR data format. This version supports NBioAPI_FIR_FORMAT_STANDARD only.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

GetHeaderFromHandle

```
public System.UInt32 GetHeaderFromHandle ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR hFIR ,  
                                           out NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_HEADER Header );  
  
public System.UInt32 GetHeaderFromHandle ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR hFIR ,  
                                           out NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_HEADER Header ,  
                                           NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_FORMAT Format);
```

Description

This function is to retrieve the FIR header information from the FIR handle. Note that this function has two different types.

Parameters

hFIR:

The handle of the FIR.

Header:

FIR header

Format:

FIR data format. This version supports NBioAPI_FIR_FORMAT_STANDARD only.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

GetTextFIRFromHandle

```
public System.UInt32 GetTextFIRFromHandle ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR hFIR ,  
                                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE TextFIR ,  
                                             System.Boolean blsWide );  
  
public System.UInt32 GetTextFIRFromHandle ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR hFIR ,  
                                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE TextFIR ,  
                                             System.Boolean blsWide ,  
                                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_FORMAT Format );
```

Description

This function is to retrieve the text-encoded FIR from the FIR handle. Note that this function has two different types.

Parameters

hFIR:

The handle of the FIR.

TextFIR:

The class that receives the text encoded FIR.

blsWide:

Flag whether an application uses Unicode characters or not.

Format:

FIR data format. This version supports NBioAPI_FIR_FORMAT_STANDARD only.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

A.1.3 BSP Method

Capture

```
public System.UInt32 Capture ( out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR );

public System.UInt32 Capture ( out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,
                               System.Int32 Timeout ,
                               NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );

public System.UInt32 Capture ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PURPOSE Purpose ,
                               out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,
                               System.Int32 Timeout ,
                               NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,
                               NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );
```

Descriptions

This method captures samples for the purpose specified, and returns the handle of FIR data. The purpose is recorded in the header of the CapturedFIR. If AuditData is non-NULL, a FIR of type “raw” may be returned. Note that this method has three different types.

Parameters

CapturedFIR:

A handle to a FIR containing captured data. This data is either an “intermediate” type FIR (which can only be used by either the Process or CreateTemplate functions, depending on the purpose), or a “processed” FIR, (which can be used directly by VerifyMatch, depending on the purpose).

Purpose:

A value indicating the purpose of the fingerprint data capture.

Timeout:

An integer specifying the timeout value (in milliseconds) for the operation. If this timeout reached, the function returns an error, and no results. This value can be any positive number. Use a value of NBioAPI.Type.TIMEOUT.DEFAULT for default timeout, and use NBioAPI.Type.TIMEOUT.INFINITE for no timeout.

AuditData:

A handle to a FIR containing raw fingerprint data. This data may be used to provide human-identifiable data of the person at the device. If the object is NULL on input, no audit data is collected.

WindowOption:

A object to a class containing the window options of the NBioBSP module.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.DEVICE_NOT_OPENED: Device not opened.

NBioAPI.Error.USER_CANCEL: Cancelled by user during verification.

Process

```
public System.UInt32 Process ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,  
                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR ProcessedFIR );  
  
public System.UInt32 Process ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,  
                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR ProcessedFIR );  
  
public System.UInt32 Process ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODING CapturedFIR ,  
                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR ProcessedFIR );  
  
public System.UInt32 Process ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR ,  
                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR ProcessedFIR );
```

Descriptions

This method processes the intermediate data captured via a call to Capture for the purpose of either verification or identification. If the processing capability is in this module, it builds a “processed” FIR, otherwise, ProcessedFIR is set to NULL. A call to this method is not necessary because the Capture and Enroll methods perform processing operation. It can be used when processing the “raw” data included from audit data.

Parameters

CapturedFIR:

The captured FIR of its handle.

ProcessedFIR:

A handle for the newly constructed “processed” FIR.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

NBioAPI.Error.ALREADY_PROCESSED: FIR already processed.

NBioAPI.Error.DATA_PROCESS_FAIL: Processing failed.

NBioAPI.Error.ENCRYPTED_DATA_ERROR: Cannot be decrypted.

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL: Data forged.

CreateTemplate

[illegible]

Descriptions

This method takes a FIR containing raw fingerprint data for the purpose of creating a new enrollment template. A new FIR is constructed from the CapturedFIR, and (optionally) it may perform an adaptation based on an existing StoredTemplate. The old StoredTemplate remains unchanged. If the StoredTemplate contains a payload, the payload is not copied into the NewTemplate. If the NewTemplate needs a payload, then that Payload must be presented as an argument to the function.

Parameters

CapturedFIR:

Object to the captured FIR. The CapturedFIR acquired for the purpose of "verification" can be used for this parameter.

StoredTemplate:

Optionally, the template to be adapted.

NewTemplate:

A handle to a newly created template that is derived from the CapturedFIR and (optionally) the StoredTemplate.

Payload:

A object to data that will be wrapped inside the newly created template. If NULL, this parameter is ignored.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

NBioAPI.Error.ENCRYPTED_DATA_ERROR: Cannot be decrypted.

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL: Data forged.

NBioAPI.Error.MUST_BE_PROCESSED_DATA: Not processed data.

VerifyMatch

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODER StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODER StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODER CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODER CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODER CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODER StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );
```


Descriptions

This function performs a verification (1-to-1) match between FIRs; the ProcessedFIR and the StoredTemplate. The ProcessedFIR is the “processed” FIR constructed specifically for the verification. The StoredTemplate was created at enrollment. If the StoredTemplate contains a Payload, the Payload may be returned upon successful verification.

Parameters

CapturedFIR:

The FIR to be verified, or its handle.

StoredTemplate:

The FIR to be verified against.

Result:

A object to a Boolean value indicating (NBioAPI.TRUE/ NBioAPI.FALSE) whether the FIRs matched or not.

Payload:

If the StoredTemplate contains a payload, it is returned in an allocated NBioAPI.TypeFIR_PAYLOAD structure.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

NBioAPI.Error.ENCRYPTED_DATA_ERROR: Cannot be decrypted.

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL: Data forged.

NBioAPI.Error.MUST_BE_PROCESSED_DATA: Not processed data.

VerifyMatchEx

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,  
                                   out System.Boolean Result ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODING CapturedFIR ,  
                                  NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODING StoredTemplate ,  
                                  out System.Boolean Result ,  
                                  NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                  NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

```
public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR ,  
                                  NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR StoredTemplate ,  
                                  out System.Boolean Result ,  
                                  NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                                  NITGEN.SDK.NBioBSP.NBioAPI.Type.MATCH_OPTION_0 MatchOption0 );
```

Descriptions

This method performs a verification (1-to-1) match between FIRs; the ProcessedFIR and the StoredTemplate. The ProcessedFIR is the “processed” FIR constructed specifically for the verification. The StoredTemplate was created at enrollment. If the StoredTemplate contains a Payload, the Payload may be returned upon successful verification. Only difference with the VerifyMatch method is that it takes a set of matching condition as a parameter

Parameters

CapturedFIR:

The FIR to be verified, or its handle.

StoredTemplate:

The FIR to be verified against.

Result:

A object to a Boolean value indicating (NBioAPI.TRUE/ NBioAPI.FALSE) whether the FIRs matched or not.

Payload:

If the StoredTemplate contains a payload, it is returned in an allocated NBioAPI_FIR_PAYLOAD structure.

MatchOption:

A set of condition for matching operation.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

NBioAPI.Error.ENCRYPTED_DATA_ERROR: Cannot be decrypted.

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL: Data forged.

NBioAPI.Error.MUST_BE_PROCESSED_DATA: Not processed data.

Enroll

```
public System.UInt32 Enroll ( out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );  
  
public System.UInt32 Enroll ( ref NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,  
                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                             System.Int32 Timeout ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );  
  
public System.UInt32 Enroll ( ref NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,  
                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                             System.Int32 Timeout ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );  
  
public System.UInt32 Enroll ( ref NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODING StoredTemplate ,  
                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                             System.Int32 Timeout ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );  
  
public System.UInt32 Enroll ( ref NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR StoredTemplate ,  
                             out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,  
                             System.Int32 Timeout ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,  
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );
```

Descriptions

This method captures fingerprint data from the attached device to create a ProcessedFIR for the purpose of enrollment.

Parameters

StoredTemplate:

Optionally, the FIR to be adapted.

NewTemplate:

A handle to a newly created template that is derived from the new raw samples and (optionally) the StoredTemplate.

Payload:

A object to data that will be wrapped inside the newly created template. If NULL, this parameter is ignored.

Timeout:

An integer specifying the timeout value (in milliseconds) for the operation. If this timeout is reached, the function returns an error, and no results. This value can be any positive number. -1 value means the FIR's default timeout value will be used.

AuditData:

A handle to a FIR containing fingerprint audit data. This data may be used to provide a human-identifiable data of the person at the device. If the object is NULL on input, no audit data is collected.

WindowOption:

A object to a class containing the window options of the NBioBSP module.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

NBioAPI.Error.DEVICE_NOT_OPENED: Device not opened.

NBioAPI.Error.USER_CANCEL: Cancelled by user during verification.

NBioAPI.Error.ENCRYPTED_DATA_ERROR: Cannot be decrypted.

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL: Data forged.

NBioAPI.Error.FUNCTION_FAIL: Data conversion failed.

Verify

```
public System.UInt32 Verify ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,
                             out System.Boolean Result ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 Verify ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,
                             out System.Boolean Result ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 Verify ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE StoredTemplate ,
                             out System.Boolean Result ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 Verify ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR StoredTemplate ,
                             out System.Boolean Result ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 Verify ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,
                             out System.Boolean Result ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,
                             System.Int32 Timeout ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );

public System.UInt32 Verify ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,
                             out System.Boolean Result ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,
                             System.Int32 Timeout ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );

public System.UInt32 Verify ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE StoredTemplate ,
                             out System.Boolean Result ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,
                             System.Int32 Timeout ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );

public System.UInt32 Verify ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR StoredTemplate ,
                             out System.Boolean Result ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload ,
                             System.Int32 Timeout ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditData ,
                             NITGEN.SDK.NBioBSP.NBioAPI.Type.WINDOW_OPTION WindowOption );
```

Descriptions

This function captures fingerprint data from the attached device, and compares it against the StoredTemplate.

Parameters

StoredTemplate:

The FIR to be verified against.

Result :

A object to a Boolean value indicating (NBioAPI.TRUE/ NBioAPI.FALSE) whether the FIRs matched or not.

Payload:

If the StoredTemplate contains a payload, it is returned in an allocated NBioAPI_FIR_PAYLOAD structure.

Timeout:

An integer specifying the timeout value (in milliseconds) for the operation. If the timeout is reached, the function returns an error, and no results. This value can be any positive number. -1 value means the FIR's default timeout value will be used.

AuditData:

A handle to a FIR containing raw fingerprint data. This data may be used to provide human-identifiable data of the person at the device. If the object is NULL on input, no audit data is collected.

WindowOption:

A object to a class containing the window options of the NBioBSP module.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_HANDLE: Invalid handle.

NBioAPI.Error.USER_CANCEL: Cancelled by user during verification.

NBioAPI.Error.ENCRYPTED_DATA_ERROR: Cannot be decrypted.

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL: Data forged.

A.1.4 User Interface Method

SetSkinResource

```
public System.Boolean SetSkinResource ( System.String szResPath );
```

Descriptions

This function is used to apply a new skin resource to the NBioBSP module. The skin resource can be made for OEM users.

Parameters

szResPath:

A fullpath of the skin resource file. If NULL, the default resource is used.

Return Value

NBioAPI.Type.TRUE: Success.

NBioAPI.Type.FALSE: Failed to find the resource DLLs. Default resource will be used.

A.1.5 NFIQ Method

GetNFIQInfo

```
public System.UInt32 GetNFIQInfo(  
    NBioAPI.Export.EXPORT_AUDIT_DATA exportAuditData  
    , out NBioAPI.Type.QUALITY_INFO_0 qualityInfo  
);
```

Descriptions

Get NFIQ data using Audit information

Parameters

exportAuditData: [in] Structure which Audit information is stored in.
qualityInfo: [out] Class to store NFIQ value.

Return Value

NBioAPI.Error.NONE : Function execution successful.

NBioAPI.Error.INVALID_HANDLE: Invalid NBioAPI handle

NBioAPI.Error.INVALID_POINTER: Invalid parameter.

GetNFIQInfoFromHandle

```
public System.UInt32 GetNFIQInfoFromHandle(  
    NBioAPI.Type.HFIR auditFIR  
    , out NBioAPI.Type.QUALITY_INFO_0 qualityInfo);
```

Descriptions

Get NFIQ data using FIR handle.

Parameters

auditFIR: [in] FIR handle class

qualityInfo: [out] Class to store NFIQ value.

Return Value

NBioAPI.Error.NONE : Function execution successful.
NBioAPI.Error.INVALID_HANDLE: Invalid NBioAPI handle.
NBioAPI.Error.INVALID_POINTER: Invalid parameter.

GetNFIQInfoFromRaw

```
public System.UInt32 GetNFIQInfoFromRaw(  
    System.Byte[] RawImage  
    , System.Int32 ImgWidth  
    , System.Int32 ImgHeight  
    , out System.Int32 NFIQ);
```

Descriptions

Get NFIQ data using RAW image.

Parameters

RawImage: [in] Byte array which Raw Image data is stored in
ImgWidth: [in] Width of image
ImgHeight: [in] Height of image
NFIQ: [out] Variable to store NFIQ value

Return Value

NBioAPI.Error.NONE : Function execution successful
NBioAPI.Error.INVALID_HANDLE: Invalid NBioAPI handle
NBioAPI.Error.INVALID_POINTER: Invalid parameter

A.1.6 Image Convert Method

Image Convert

```
public System.UInt32 ImgConvRawToBmpBuf(  
    byte[] inBuffer,  
    System.UInt32 nWidth,  
    System.UInt32 nHeight,  
    out byte[] outBuffer)  
  
public System.UInt32 ImgConvRawToBmpBufEx(  
    byte[] inBuffer,  
    System.UInt32 nWidth,  
    System.UInt32 nHeight,  
    System.UInt32 nDPI,  
    out byte[] outBuffer)  
  
public System.UInt32 ImgConvBmpToRawBuf(  
    byte[] inBuffer,  
    out System.UInt32 nWidth,  
    out System.UInt32 nHeight,  
    out byte[] outBuffer);  
  
public System.UInt32 ImgConvRawToJpgBuf(  
    byte[] inBuffer,  
    System.UInt32 nWidth,
```

```
System.UInt32 nHeight,  
System.UInt32 nQuality,  
out byte[] outBuffer);
```

```
public System.UInt32 ImgConvRawToJpgBufEx(  
    byte[] inBuffer,  
    System.UInt32 nWidth,  
    System.UInt32 nHeight,  
    System.UInt32 nQuality,  
    System.UInt32 nDPI,  
    out byte[] outBuffer);
```

```
public System.UInt32 ImgConvJpgToRawBuf(  
    byte[] inBuffer,  
    out System.UInt32 nWidth,  
    out System.UInt32 nHeight,  
    out byte[] outBuffer);
```

```
public System.UInt32 ImgConvRawToWSQBuf(  
    byte[] inBuffer,  
    System.UInt32 nWidth,  
    System.UInt32 nHeight,  
    out byte[] outBuffer,  
    float p);
```

```
public System.UInt32 ImgConvWSQToRawBuf(  
    byte[] inBuffer,  
    out System.UInt32 nWidth,  
    out System.UInt32 nHeight,  
    out byte[] outBuffer);
```

Descriptions

Convert from RAW image to BMP, Jpeg, WSQ formats.

Parameters

inBuffer: [in] Byte array which initial image data is stored in
nWidth: [in] The number of pixels indicating the width of the image (pixel of HFDU01: 248)
nHeight: [in] The number of pixels indicating the height of the image (pixel of HFDU01: 292)
nQuality: [in] If compression ratio is set as high as 100:1, Jpeg file will be larger than its RAW file counterparts.
nDPI: [in] DPI(NITGEN&COMPANY Image Sensor : 500DPI) of RAW image
outBuffer: [out] Byte array to store converted image data

Return Value

NBioAPI.Error.NONE : Function execution successful
NBioAPI.Error.INVALID_HANDLE: Invalid NBioAPI handle
NBioAPI.Error.INVALID_POINTER: Invalid parameter
NBioAPI.Error.IMGCONV_INVALID_PARAM: Invalid parameter
NBioAPI.Error.IMGCONV_MEMALLOC_FAIL: Memory allocation error

A.1.7 ISO 19794-4 Method/Class/Struct

ISO 19794-4 Export Method

```
public static System.UInt32 ExportRawToISOV1(
    NBioAPI.Export.EXPORT_AUDIT_DATA ExportAuditData,
    System.Boolean blsRollDevice,
    System.Byte nCompressMod,
    out System.Byte[] ISOBuf);
```

```
public static System.UInt32 ExportRawToISOV2(
    System.Byte nFingerID,
    System.UInt16 nImgWidth,
    System.UInt16 nImgHeight,
    System.Byte[] RawBuf,
    System.Boolean blsRollDevice,
    System.Byte nCompressMod,
    out System.Byte[] ISOBuf);
```

```
public static System.UInt32 ExportRawToANSIV1(
    NBioAPI.Export.EXPORT_AUDIT_DATA ExportAuditData,
    System.Boolean blsRollDevice,
    System.Byte nCompressMod,
    out System.Byte[] ISOBuf);
```

```
public static System.UInt32 ExportRawToANSIV2(
    System.Byte nFingerID,
    System.UInt16 nImgWidth,
    System.UInt16 nImgHeight,
    System.Byte[] RawBuf,
    System.Boolean blsRollDevice,
    System.Byte nCompressMod,
    out System.Byte[] ISOBuf);
```

Descriptions

Convert from image data to ISO 19794-4 data.

Parameters

ExportAuditData: [in] Image structure from NBioAPI.Export.NBioBSPTolImage method
 blsRollDevice: [in] Specify whether input is Roll Image. (true: Roll Image, false: Flat Image)
 nCompressMod: [in] Select compression mode. (Refer to COMPRESS_MOD)
 nFingerID: [in] Image structure from NBioAPI.Export.NBioBSPTolImage method
 nImgWidth: [in] Width of Raw image
 nImgHeight: [in] Height of Raw image
 RawBuf: [in] Data array of Raw image
 ISOBuf: [out] ISO 19794-4 data array

Return Value

NBioAPI.Error.NONE : Function execution successful
 NBioAPI.Error.INVALID_HANDLE: Invalid NBioAPI handle
 NBioAPI.Error.INVALID_POINTER: Invalid parameter
 NBioAPI.Error.FUNCTION_FAIL: Conversion failure

ISO 19794-4 Import Method

```
public static System.UInt32 ImportISOToRaw(  
    System.Byte[]          ISOBuf,  
    out NIMPORTRAWSET      ImportRawSet)  
  
public static System.UInt32 ImportANSIToRaw(  
    System.Byte[]          ISOBuf,  
    out NIMPORTRAWSET      ImportRawSet)
```

Descriptions

Convert from ISO 19794-4 data to image information.

Parameters

ISOBuf: [in] ISO 19794-4 data array

ImportRawSet: [in] Raw Images stored in ISOBuf are returned. (Refer to NIMPORTRAWSET)

Return Value

NBioAPI.Error.NONE : Fuction execution successful

NBioAPI.Error.INVALID_HANDLE: Invalid NBioAPI handle

NBioAPI.Error.INVALID_POINTER: Invalid parameter

NBioAPI.Error.FUNCTION_FAIL: Conversion failure

ISO 19794-4 Class/Struct

```
public class COMPRESS_MOD  
{  
    public const System.Byte NONE   = (0);  
    public const System.Byte WSQ    = (1);  
}
```

Descriptions

Constant value which specifies whether Raw Image is compressed

Attribute

NONE: Do not compress image.

WSQ: Compress image in WSQ format

```
public struct NIMPORTRAW  
{  
    public System.Byte    nFingerID;  
    public System.UInt16  nImgWidth;  
    public System.UInt16  nImgHeight;  
    public System.Byte[]  RawData;  
}
```

Descriptions

Structure to store Raw Image.

Attribute

nFingerID: Assign finger identification number. (Refer to NBioAPI_FINGER_ID of eNBioBSP SDK)

nImgWidth: Assign the width of image

nImgHeight: Assign the height of image

RawData: Image data

```
public struct NIMPORTRAWSET
```

```
{
```

```
    public System.Byte    nDataCount;
```

```
    public NIMPORTRAW[]  ImportRawData;
```

```
}
```

Descriptions

Structure to store a Raw Image set

Attribute

nDataCount: Assign the number of data

ImportRawData: NIMPORTRAW data array

A.2 NBioAPI.Export Class

This class includes some data conversion functions.

Export

```
public Export (NITGEN.SDK.NBioBSP.NBioAPI NBioBSP );
```

Descriptions

This function takes an NBioAPI class to be constructed.

Parameters

NBioBSP:

An NBioAPI class.

Example

```
m_NBioAPI = new NBioAPI();  
m_Export = new NBioAPI.Export(m_NBioAPI);
```

FDxToNBioBSP

```
public System.UInt32 FDxToNBioBSP ( byte[] FDxData ,  
                                     NBioAPI.Type.MINCONV_DATA_TYPE FDxDataType ,  
                                     NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PURPOSE Purpose ,  
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR ProcessedFIR )
```

Descriptions

This function is to convert the FDx data (400-byte minutiae array) to the FIR format in a handle of FIR. This function does not accept raw data.

Parameters

FDxData:

The FDxData to be converted.

FDxDataType :

The type of the FDxData.

Purpose:

A value indicating the desired purpose of the ProcessedFIR.

ProcessedFIR:

The ProcessedFIR handle.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_MINSIZE: Invalid size of minutiae data.

NBioAPI.Error.FUNCTION_FAIL: Conversion failed.

FDxToNBioBSPEX

```
public System.UInt32 FDxToNBioBSPEX ( byte[] FDxData ,  
                                     UInt32 FDxTemplateSize,  
                                     NBioAPI.Type.MINCONV_DATA_TYPE FDxDatatype ,  
                                     NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PURPOSE Purpose ,  
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR ProcessedFIR )
```

Descriptions

This function is to convert the FDx data to the FIR format in a handle of FIR. This function does not accept raw data.

Parameters

FDxData:

The FDxData to be converted.

FDxTemplateSize:

The size of a single minutiae data

FDxDatatype :

The type of the FDxData.

Purpose:

The purpose of ProcessedFIR

ProcessedFIR:

A value indicating the desired purpose of the ProcessedFIR.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_MINSIZE: Invalid size of minutiae data.

NBioAPI.Error.FUNCTION_FAIL: Conversion failed.

NBioBSPToFDx

```
public System.UInt32 NBioBSPToFDx ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,  
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_DATA ExportData ,  
                                     NITGEN.SDK.NBioBSP.NBioAPI.Type.MINCONV_DATA_TYPE ExportType );
```

```
public System.UInt32 NBioBSPToFDx ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,  
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_DATA ExportData ,  
                                     NITGEN.SDK.NBioBSP.NBioAPI.Type.MINCONV_DATA_TYPE ExportType );
```

```
public System.UInt32 NBioBSPToFDx ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR ,  
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_DATA ExportData ,  
                                     NITGEN.SDK.NBioBSP.NBioAPI.Type.MINCONV_DATA_TYPE ExportType );
```

```
public System.UInt32 NBioBSPToFDx ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR ,  
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_DATA ExportData ,  
                                     NITGEN.SDK.NBioBSP.NBioAPI.Type.MINCONV_DATA_TYPE ExportType );
```

Descriptions

This function is to convert the FIR data to FDx data format.

Parameters

CapturedFIR:

The FIR data to be converted. It can take a FIR handle, binary FIR or text encoded FIR.

ExportData:

A object to a NBioAPI.Export.EXPORT_DATA class that receives the data converted from the FIR input.

FDxDataType :

A value indicating the type of exportation.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.MUST_BE_PROCESSED_DATA: Not processed data.

NBioAPI.Error.UNKNOWN_FORMAT: Unknown format.

NBioBSPToImage

```
public System.UInt32 NBioBSPToImage ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR AuditFIR ,
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_AUDIT_DATA ExportAuditData );

public System.UInt32 NBioBSPToImage ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR AuditFIR ,
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_AUDIT_DATA ExportAuditData );

public System.UInt32 NBioBSPToImage ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE AuditFIR ,
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_AUDIT_DATA ExportAuditData );

public System.UInt32 NBioBSPToImage ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR AuditFIR ,
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_AUDIT_DATA ExportAuditData );
```

Descriptions

This function is to convert the FIR data to an image data.

Parameters

AuditFIR:

A object to a class specifying the FIR to be converted to image data. It can take a FIR handle, binary FIR or text encoded FIR.

ExportAuditData:

A object to a NBioAPI.Export.EXORT_DATA class that receives the image data converted from the FIR. This structure contains information about the FIR and the raw image data.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.ALREADY_PROCESSED: FIR already processed.

ImageToNBioBSP

```
public System.UInt32 ImageToNBioBSP ( NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_AUDIT_DATA ExportAuditData
                                     out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR FIR );
```

Descriptions

This function is to convert raw image data into a FIR format.

Parameters

ExportAuditData:

A obejct to a NBioAPI.Export.EXORT_DATA class that receives the image data converted from the FIR. This structure contains information about the FIR and the raw image data.

FIR:

A object to a HFIR that receives the FIR handle.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_MINSIZE: Invalid size of minutiae data.

ImportDataToNBioBSP

```
public System.UInt32 ImportDataToNBioBSP ( NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_DATA ExportData ,  
                                           NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PURPOSE Purpose ,  
                                           out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR ProcessedFIR );
```

Descriptions

This function is to convert minutiae data into a FIR format.

Parameters

ExportData:

A object to a NBioAPI.Export.EXPORT_DATA class to be converted.

Purpose:

A value indicating the purpose of conversion.

ProcessedFIR:

A object to a HFIR that receives the FIR handle.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_MINSIZE: Invalid size of minutiae data.

ImportDataToNBioBSPEx

```
public System.UInt32 ImportDataToNBioBSPEx ( NITGEN.SDK.NBioBSP.NBioAPI.Export.EXPORT_DATA ExportData ,  
                                              NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PURPOSE Purpose ,  
                                              NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_DATA_TYPE DataType ,  
                                              out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR ProcessedFIR );
```

Descriptions

This function is to convert minutiae data into a FIR format.

Parameters

ExportData:

A object to a NBioAPI.Export.EXPORT_DATA class to be converted.

Purpose:

A value indicating the purpose of conversion.

DataType:

A type of minutiae data. (RAW, PROCESSED, ...)

ProcessedFIR:

A object to a HFIR that receives the FIR handle.

Return Value

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INVALID_MINSIZE: Invalid size of minutiae data.

A.3 NBioAPI.IndexSearch Class

This class includes the IndexSearch functions.

A.3.1 Initialization Method

IndexSearch

```
public System.UInt32 IndexSearch (NITGEN.SDK.NBioBSP.NBioAPI NBioBSP );
```

Descriptions

This function is a constructor of the IndexSearch class. It takes the NBioAPI class.

Parameters

NBioBSP:

The NBioAPI class.

Example

```
m_NBioAPI = new NBioAPI();  
m_IndexSearch = new NBioAPI.IndexSearch(m_NBioAPI);
```

InitEngine

```
public System.UInt32 InitEngine ( );
```

Descriptions

This function is to initialize the IndexSearch engine. It allocates the memory for fingerprint DB and initializes global variables.

Parameters

N/A

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

TerminateEngine

```
public System.UInt32 TerminateEngine ( );
```

Descriptions

This function is to close the IndexSearch engine. It must be called before an application is closed to free all memory allocated for the IndexSearch engine.

Parameters

N/A.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

GetInitInfo

```
public System.UInt32 GetInitInfo ( out NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.INIT_INFO_0 InitInfo0 );
```

Descriptions

This function is to receive the current parameter values of the IndexSearch engine.

Parameters

InitInfo0

A object to NBioAPI.IndexSearch.INIT_INFO_0 that receives the initialization setting for the IndexSearch engine.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

SetInitInfo

```
public System.UInt32 SetInitInfo ( out NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.INIT_INFO_0 InitInfo0 );
```

Descriptions

This function is to configure the parameter values of the IndexSearch engine.

Parameters

InitInfo0

A object to NBioAPI.IndexSearch.INIT_INFO_0 containing the initialization setting for the IndexSearch engine.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

NBioAPI.Error.INIT_PRESEARCHRATE: Invalid value for PreSearchRate.

A.3.2 Enroll / Remove / Search Method

AddFIR

```
public System.UInt32 AddFIR ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR,  
                             System.UInt32 UserID,  
                             NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO[] SampleInfo );  
  
public System.UInt32 AddFIR ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR,  
                             System.UInt32 UserID,  
                             NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO[] SampleInfo );  
  
public System.UInt32 AddFIR ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR,  
                             System.UInt32 UserID,  
                             NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO[] SampleInfo );  
  
public System.UInt32 AddFIR ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR,  
                             System.UInt32 UserID,  
                             NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO[] SampleInfo );
```

Descriptions

This function is to register a fingerprint template data, along with a user ID, into the fingerprint DB on memory. After successful registration, it returns the template information.

Parameters

CapturedFIR

A FIR data to be registered.

UserID

A user ID number to be registered.

SampleInfo

A object to a NBioAPI.IndexSearch.FP_INFO that receives some information of a registered template, including finger IDs and sample counts.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

RemoveData

```
public System.UInt32 RemoveData ( NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO FpInfo );
```

```
public System.UInt32 RemoveData ( System.UInt32 UserID,  
                                  System.Byte FingerID,  
                                  System.Byte SampleNumber );
```

Descriptions

This function is to remove a template data from the fingerprint DB on memory. The parameter, pFpInfo, is used to search the corresponding template data.

Parameters

FpInfo

A object to a NBioAPI.IndexSearch.FP_INFO containing template information including a user ID, finger IDs and sample numbers.

UserID

A user ID number to be deleted.

FingerID

A finger ID to be deleted.

SampleNumber

A sample number to be deleted.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

RemoveUser

```
public System.UInt32 RemoveUser ( System.UInt32 nUserID );
```

Descriptions

This function is to remove all template data of a user from the fingerprint DB on memory.

Parameters

UserID

A user ID number to be deleted.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

IdentifyData

```
public System.UInt32 IdentifyData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR,
                                   System.UInt32 SecuLevel,
                                   out NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO FpInfo,
                                   NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.CALLBACK_INFO_0 CallbackInfo);

public System.UInt32 IdentifyData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR,
                                   System.UInt32 SecuLevel,
                                   out NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO FpInfo,
                                   NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.CALLBACK_INFO_0 CallbackInfo );

public System.UInt32 IdentifyData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR,
                                   System.UInt32 SecuLevel,
                                   out NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO FpInfo,
                                   NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.CALLBACK_INFO_0 CallbackInfo );

public System.UInt32 IdentifyData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR,
                                   System.UInt32 SecuLevel,
                                   out NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO FpInfo,
                                   NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.CALLBACK_INFO_0 CallbackInfo );
```

Descriptions

This function is to perform identification and determine if the same fingerprint exists within the fingerprint DB. After identification, it returns the template information if successful. The callback functions entered through the last parameter, pCallbackInfo0, can be used to determine to skip or stop verification. For more information, please refer to NBioAPI_INDEXSEARCH_CALLBACK_INFO_0 structure.

Parameters

CapturedFIR

A FIR data.

SecuLevel

Indicates the security level set for fingerprint recognition. Values range from 1 (lowest) to 9 (highest). The default is 5 (normal).

FpInfo

A object to a NBioAPI.IndexSearch.FP_INFO that receives template information.

CallbackInfo

A object to a NBioAPI.IndexSearch.CALLBACK_INFO_0 containing a set of pointers of callback functions that is to invoked during IndexSearch operation. If NULL, no callback function will be used.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

NBioAPI.Error.INDEXSEARCH_IDENTIFY_STOP: Identification stopped by a callback function.

A.3.2 DB Method

SaveDBToFile

```
public System.UInt32 SaveDBToFile ( System.String szFilepath );
```

Descriptions

This function is to backup the fingerprint DB, in memory, into a file. It takes a full path and creates a file containing fingerprint DB.

Parameters

szFilepath

Location and file name to make a DB file.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

NBioAPI.Error.INDEXSEARCH_SAVE_DB: DB file not saved.

LoadDBFromFile

```
public System.UInt32 LoadDBFromFile ( System.String szFilepath );
```

Descriptions

This function is to load the fingerprint DB file into memory.

Parameters

szFilepath

Location and file name to load a DB file.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

NBioAPI.Error.INDEXSEARCH_LOAD_DB: DB file not loaded.

ClearDB

```
public System.UInt32 ClearDB ( );
```

Descriptions

This function is to delete all template data from the fingerprint DB in memory.

Parameters

N/A.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

GetDataCount

```
public System.UInt32 GetDataCount (out System.UInt32 DataCount);
```

Descriptions

This function is to retrieve the count of template data in the fingerprint DB.

Parameters

DataCount

The count of template data stored in the fingerprint DB.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

CheckDataExist

```
public System.UInt32 CheckDataExist ( NITGEN.SDK.NBioBSP.NBioAPI.IndexSearch.FP_INFO FpInfo,  
                                     out System.Boolean      Exist);
```

Descriptions

This function is to check if a specific template data exists in the fingerprint DB.

Parameters

FpInfo

A object to template information.

Exist

A object to a Boolean that receives the flag of existence.

Return Values

NBioAPI.Error.NONE: No error.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL: Engine not initialized.

A.4 NBioAPI.NSearch Class

This class includes the NSearchSearch functions.

A.4.1 Initialization Method

NSearch

```
public System.UInt32 NSearch (NITGEN.SDK.NBioBSP.NBioAPI NBioBSP );
```

Descriptions

This function is a constructor of the IndexSearch class. It takes the NBioAPI class.

Parameters

NBioBSP:

The NBioAPI class.

Example

```
m_NBioAPI = new NBioAPI();  
m_NSearch = new NBioAPI.NSearch(m_NBioAPI);
```

InitEngine

```
public System.UInt32 InitEngine ( );
```

Descriptions

This function is to initialize the NSearch engine. It allocates the memory for fingerprint DB and initializes global variables.

Parameters

N/A.

Return Values

NBioAPI.Error.NONE : No error

NBioAPI.Error.NSEARCH_OPEN_FAIL : Engine not opened.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

TerminateEngine

```
public System.UInt32 TerminateEngine ( );
```

Descriptions

This function is to close the IndexSearch engine. It must be called before an application is closed to free all memory allocated for the NSearch engine.

Parameters

N/A.

Return Values

NBioAPI.Error.NONE : No error

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

GetInitInfo

```
public System.UInt32 GetInitInfo ( out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.INIT_INFO_0 InitInfo0 );
```

Descriptions

This function is to receive the current parameter values of the NSearch engine.

Parameters

InitInfo0

A object to NBioAPI.NSearch.INIT_INFO_0 that receives the initialization setting for the NSearch engine.

Return Values

NBioAPI.Error.NONE : No Error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

SetInitInfo

```
public System.UInt32 SetInitInfo ( out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.INIT_INFO_0 InitInfo0 );
```

Descriptions

This function is to configure the parameter values of the NSearch engine.

Parameters

InitInfo0

A object to NBioAPI.NSearch.INIT_INFO_0 containing the initialization setting for the IndexSearch engine.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

NBioAPI.Error.INIT_MAXCANDIDATE : Invalid value for MaxCandidate.

A.4.2 Enroll / Remove / Search Method

AddFIR

```
public System.UInt32 AddFIR ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR,  
                             System.UInt32 UserID,  
                             NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO[] SampleInfo );  
  
public System.UInt32 AddFIR ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR,  
                             System.UInt32 UserID,  
                             NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO[] SampleInfo );  
  
public System.UInt32 AddFIR ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODING CapturedFIR,  
                             System.UInt32 UserID,  
                             NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO[] SampleInfo );  
  
public System.UInt32 AddFIR ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR,  
                             System.UInt32 UserID,  
                             NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO[] SampleInfo );
```

Descriptions

This function is to register a fingerprint template data, along with a user ID, into the fingerprint DB on memory. After successful registration, it returns the template information.

Parameters

CapturedFIR

A FIR data to be registered.

UserID

A user ID number to be registered.

SampleInfo

A object to a NBioAPI.NSearch.FP_INFO that receives some information of a registered template, including finger IDs and sample counts.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

NBioAPI.Error.NSEARCH_OVER_LIMIT : Over template count

NBioAPI.Error.NSEARCH_MEM_OVERFLOW : Out of memory.

RemoveData

```
public System.UInt32 RemoveData ( NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO FpInfo );
```

```
public System.UInt32 RemoveData ( System.UInt32 UserID,  
                                  System.Byte FingerID,  
                                  System.Byte SampleNumber );
```

Descriptions

This function is to remove a template data from the fingerprint DB on memory. The parameter, pFpInfo, is used to search the corresponding template data.

Parameters

FpInfo

A object to a NBioAPI.NSearch.FP_INFO containing template information including a user ID, finger IDs and sample numbers.

UserID

A user ID number to be deleted.

FingerID

A finger ID to be deleted.

SampleNumber

A sample number to be deleted.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

RemoveUser

```
public System.UInt32 RemoveUser ( System.UInt32 nUserID );
```

Descriptions

This function is to remove all template data of a user from the fingerprint DB on memory.

Parameters

UserID

A user ID number to be deleted.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

SearchData

```
public System.UInt32 SearchData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR,  
                                out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.CANDIDATE[] Candidate);  
  
public System.UInt32 SearchData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR,  
                                out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.CANDIDATE[] Candidate);  
  
public System.UInt32 SearchData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODING CapturedFIR,  
                                out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.CANDIDATE[] Candidate);  
  
public System.UInt32 SearchData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR,  
                                out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.CANDIDATE[] Candidate);
```

Descriptions

This function is to perform Search and return candidate lists.

Parameters

CapturedFIR

A FIR data.

Candidate

A object to a NBioAPI.NSearch.CANDIDATE that receives some information of a searched template.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

IdentifyData

```
public System.UInt32 IdentifyData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR,  
                                   System.UInt32 SecuLevel,  
                                   out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO FpInfo);  
  
public System.UInt32 IdentifyData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR,  
                                   System.UInt32 SecuLevel,  
                                   out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO FpInfo);  
  
public System.UInt32 IdentifyData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR,  
                                   System.UInt32 SecuLevel,  
                                   out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO FpInfo);  
  
public System.UInt32 IdentifyData ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR,  
                                   System.UInt32 SecuLevel,  
                                   out NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO FpInfo);
```

Descriptions

This function is to perform identification and determine if the same fingerprint exists within the fingerprint DB. After identification, it returns the template information if successful.

Parameters

CapturedFIR

A FIR data.

SecuLevel

Indicates the security level set for fingerprint recognition. Values range from 1 (lowest) to 9 (highest). The default is 5 (normal).

FpInfo

A object to a NBioAPI.NSearch.FP_INFO that receives template information.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

NBioAPI.Error.NSEARCH_IDENTIFY_FAIL : Identify failed.

A.4.2 DB Method

SaveDBToFile

```
public System.UInt32 SaveDBToFile ( System.String szFilepath );
```

Descriptions

This function is to backup the fingerprint DB, in memory, into a file. It takes a full path and creates a file containing fingerprint DB.

Parameters

szFilepath : Location and file name to make a DB file.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

NBioAPI.Error.NSEARCH_SAVE_DB : DB file not saved.

LoadDBFromFile

```
public System.UInt32 LoadDBFromFile ( System.String szFilepath );
```

Descriptions

This function is to load the fingerprint DB file into memory.

Parameters

szFilepath : Location and file name to load a DB file.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

NBioAPI.Error.NSEARCH_LOAD_DB : DB file not loaded.

ClearDB

```
public System.UInt32 ClearDB ( );
```

Descriptions

This function is to delete all template data from the fingerprint DB in memory.

Parameters

N/A.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

GetDataCount

```
public System.UInt32 GetDataCount (out System.UInt32 DataCount);
```

Descriptions

This function is to retrieve the count of template data in the fingerprint DB.

Parameters

DataCount

The count of template data stored in the fingerprint DB.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

CheckDataExist

```
public System.UInt32 CheckDataExist ( NITGEN.SDK.NBioBSP.NBioAPI.NSearch.FP_INFO FpInfo,  
                                     out System.Boolean      Exist);
```

Descriptions

This function is to check if a specific template data exists in the fingerprint DB.

Parameters

FpInfo

A object to template information.

Exist

A object to a Boolean that receives the flag of existence.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

ImportIndexSearchDB

```
public System.UInt32 ImportIndexSearchDB ( System.String szFilepath );
```

Descriptions

This function is to load the fingerprint DB file(IndexSearch) into memory.

Parameters

szFilepath : Location and file name to load a DB file.

Return Values

NBioAPI.Error.NONE : No error.

NBioAPI.Error.NSEARCH_INIT_FAIL : Engine not initialized.

NBioAPI.Error.NSEARCH_LOAD_DB : DB file not loaded.