eNBSP - NBioBSP

NITGEN&COMPANY Biometric Service Provider SDK

Programmer's Manual .NET

SDK version 4.8x

Serial Number:

Specifications subject to change without notice.

목 차

제 1 장 개요

eNBSP (이하 NBioBSP 로 표시)는 니트젠 지문 인식기를 이용하여 어플리케이션을 개발할 수 있도록 만든 High-Level SDK 로써 니트젠에서 설계한 지문 인식을 위한 API 인 NBioAPI 를 바탕으로 제작되었다.

NBioBSP 는 기본적으로 지문 인증과 관련된 모든 API 들과 함께 사용자 등록 및 인증을 위한 사용자 인터페이스 등을 위저드 방식으로 모두 제공하기 때문에 개발자들은 최소한의 노력으로 개발코자 하는 어플리케이션에 니트젠의 지문 인식 기능을 접목시킬 수 있다.

1.1 제공 모듈

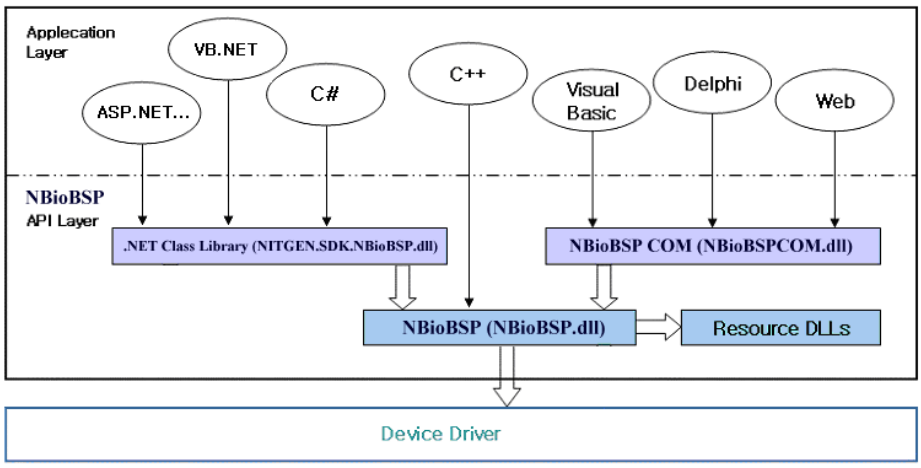
- NBioBSP.dll

  NBioBSP.dll 은 사용자 등록, 인증 등 지문 인증과 관련된 모든 기능을 구현하고 있는 NBioBSP 의 핵심 모듈이다.

- NITGEN.SDK.NBioBSP.dll

  NITGEN.SDK.NBioBSP.dll 은 Microsoft .NET 환경에서 C#, VB.NET, ASP.NET, J# 등의 .NET 언어 개발자를 지원하기 위한 목적으로 개발되었다.

  NITGEN.SDK.NBioBSP.dll 은 NBioBSP.dll 을 이용하여 개발되었으며, NBioBSP.dll 의 거의 모든 기능을 제공한다.



[NBioBSP SDK 개발 모델]

1.2 Sample 프로그램

1.2.1 제공되는 Sample (32bit SDK)

- C#

  BSPDemoCS – NBioBSP 의 기본적인 기능에 대한 예제

  UITestCS – NBioBSP 의 UI 옵션을 조정하는 기능에 대한 예제

  IndexSerchDemoCS – NBioBSP 의 IndexSearch 를 사용하는 방법에 대한 예제

  RollDemoCS – NBioBSP 의 회전 지문 취득에 대한 예제

1.2.2 제공되는 Sample (64bit SDK)

- C#

  BSPDemoCS – NBioBSP 의 기본적인 기능에 대한 예제

  UITestCS – NBioBSP 의 UI 옵션을 조정하는 기능에 대한 예제

  IndexSerchDemoCS – NBioBSP 의 IndexSearch 를 사용하는 방법에 대한 예제

  BSPRollDemoCS – NBioBSP 의 회전 지문 취득에 대한 예제

  ExportDemoCS – NBioBSP 의 데이터 Export / Import 기능에 대한 예제

- VB.NET

  BSPDemoVBNET – NBioBSP 의 기본적인 기능에 대한 예제

  IndexSerchDemoVBNET – NBioBSP 의 IndexSearch 를 사용하는 방법에 대한 예제

제 2 장 .NET 프로그래밍

이 장에서는 NITGEN.SDK.NBioBSP.dll 모듈을 이용한 C# 프로그래밍에 대해 설명한다.

2.1 모듈 초기화 및 종료

2.1.1 모듈 초기화

NBioBSP Class Library 모듈을 초기화 하는 방법은 다음과 같다.

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

2.1.2 모듈 사용 종료

.NET 언어의 특성상 종료를 위해 메모리를 해제하거나 할 필요는 없다.(.NET GC 가 동작할 때 자동으로 해제된다.)
하지만 NBioBSP.dll 의 메모리를 명시적으로 해제하고 싶을 때는 다음과 같이 할 수 있다.

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

Dispose 메소드는 NBioAPI 와 NBioAPI.Type.HFIR 만 가지고 있으며, NBioBSP.dll 에서 사용한 메모리를 바로 해제하는 기능을 한다.

2.2 디바이스 관련 프로그래밍

특정 디바이스를 사용하기 위해서는 반드시 디바이스를 오픈 하는 과정을 거쳐야 한다. 먼저 시스템에 어떤 디바이스들이 연결되어 있는지에 대한 정보를 얻기 위해서는 Enumerate 메소드를 이용하면 된다

2.2.1 디바이스 열거하기

디바이스를 오픈 하기 전에 EnumerateDevice 메소드를 이용하여 사용자의 PC 에 연결되어 있는 디바이스의 개수 및 종류를 알 수 있다. EnumerateDevice 를 호출하면 현재 시스템에 부착되어 있는 디바이스의 개수와 각 디바이스에 대한 ID 값을 얻을 수 있다. DeviceID 는 내부적으로 디바이스 이름과 인스턴스 번호로 구성되어 있다.

```
DeviceID = Instance Number + Device Name
```

만일 시스템의 각 타입의 디바이스가 하나씩 만 존재하는 경우 인스턴스 번호가 0 이므로 디바이스 이름과 DeviceID 는 같은 값을 갖게 된다.

아래는 EnumerateDevice 메소드를 사용해서 ComboBox 에 추가하는 예제를 보여주고 있다.

```
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);
        }
}
```

nDeviceID[DeviceNumber] 속성에서 DeviceNumber 부분에 디바이스 번호를 입력하면 해당 디바이스의 ID 를 알려준다. 예를 들어 첫번째 디바이스의 DeviceID 를 알고 싶은 경우엔 nDeviceID[0] 라고 써주면 된다.

2.2.2 디바이스 초기화

NBioBSP Class Library 에서 사용할 디바이스를 선택하기 위해서는 OpenDevice 메소드를 호출하면 된다. Enroll, Verify, Capture 등, 디바이스와 관련된 작업을 수행하기 위해서는 반드시 OpenDevice 메소드를 이용하여 디바이스 초기화를 먼저 수행하여야 한다.

설치되어 있는 디바이스의 ID 를 모르는 경우에는 EnumerateDevice 메소드를 이용하여 먼저 설치되어 있는 디바이스 ID 리스트를 얻는다.

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

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

만일 자동으로 사용할 디바이스를 지정하고 싶으면 DeviceID 로 NBioAPI.Type.DEVICE_ID.AUTO 를 사용하면 된다.

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

NBioAPI.Type.DEVICE_ID.AUTO 를 사용하면 디바이스가 여러 개 있는 경우 가장 최근에 사용한 디바이스를 먼저 검색하게 된다.

2.2.3 디바이스 사용 끝내기

사용 중인 디바이스를 더 이상 사용할 필요가 없을 경우 CloseDevice 메소드를 이용하여 디바이스의 사용을 해제할 수 있다. CloseDevice 메소드를 호출할 때는 반드시 Open 메소드를 호출할 때 사용했던 DeviceID 를 사용하여야 한다.

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

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

다른 디바이스를 오픈 하는 경우에도 반드시 이전에 사용하고 있던 디바이스의 사용을 먼저 해제해주어야 한다.

2.3 지문 등록

지문을 등록하기 위해서는 Enroll 메소드를 사용한다. NBioBSP Class Library 모듈에서는 모든 지문 데이터를 핸들 형태와 바이너리 형태, 그리고 텍스트 인코딩된 형태로 사용할 수 있다. Enroll 이 성공하면 FIR 에 대한 핸들 값을 얻어올 수 있고 이 핸들 값을 이용해 바이너리 형태나 텍스트 인코딩된 형태로 다시 얻어올 수 있다. NBioBSP Class Library 는 오버로딩(Overloading)된 여러 개의 Enroll 메소드를 제공하는데 필요에 따라 원하는 메소드를 사용하면 된다. 그 중 간단한 사용 예는 아래와 같다.

```
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 ...
```

지문 데이터를 저장하려면 biFIR 또는 textFIR 을 파일 또는 DB 에 저장하면 된다.

2.4 지문 인증

인증을 수행할 때는 Verify 메소드를 사용한다. Verify 메소드는 파라미터로 이전에 등록되어 있던 지문 데이터를 취한다. Verify 메소드는 현재 입력 받은 지문 데이터와 등록되어 있던 지문 데이터를 비교하며 그 결과값을 돌려준다. 만약 인증이 성공하고 이전에 등록되어 있던 지문 데이터에 Payload 가 존재한다면 Payload 값까지 돌려주게 된다.

```
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 환경에서의 프로그래밍

독립 실행형 환경과는 달리 클라이언트/서버 환경에서는 지문을 입력 받는 곳과 매칭하는 곳이 다르다. 즉 클라이언트에서는 보통 지문을 캡쳐하고 서버에서 매칭이 이루어진다. 등록을 위한 지문을 입력 받을 때는 Enroll 메소드를 사용하고 인증을 위한 지문을 입력 받을 때는 Capture 메소드를 사용한다.

서버에서 매칭을 하는 경우에는 VerifyMatch 메소드를 사용한다. VerifyMatch 메소드는 파라미터로 등록되어 있던 지문 데이터와 클라이언트로부터 입력 받은 지문을 취한다.

2.5.1 지문 등록하기

클라이언트에서 등록용 지문 데이터를 입력 받기 위해서는 Enroll 메소드를 사용한다.

```
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 지문 인증하기

먼저 인증에 사용할 지문 데이터를 클라이언트로부터 얻어온다. 이때 사용할 메소드는 Capture 이다.
Enroll 메소드와 Capture 메소드를 이용하여 가져오는 지문 데이터의 차이점은 Enroll 메소드를 이용하면 어떤 손가락을 등록했는지에 대한 정보를 가지고 있어 여러 개의 지문 정보를 하나의 지문 데이터로 전송이 가능한 반면 Capture 메소드를 이용하면 단순히 하나의 지문 데이터만 입력 받는다. Capture 메소드를 사용하기 위해서는 Extraction object 를 선언한 다음 사용하게 되며 Capture 의 용도를 인자로 넘겨 주어야 한다. 현재 인자로 사용하는 값이 여러 개 있으나 현재는 NBioAPI_FIR_PURPOSE_VERIFY 만을 지원한다.

```
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 ...
```

서버쪽에서의 저장된 지문 데이터 간의 비교는 VerifyMatch 메소드를 이용하면 된다. VerifyMatch 메소드에는 클라이언트에서 넘겨 받은 FIR 과 서버에 저장된 FIR 이렇게 두 개의 인자를 넣어 주어야 한다. VerifyMatch 메소드는 인증이 성공 했을 경우 Payload 를 되돌려준다. 이때 Payload 는 두 번째 인수인 StoredFIR 이 가지고 있던 Payload 로 첫번째 CapturedFIR 의 Payload 에는 영향을 받지 않는다.

```
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
```

2.4 Payload 사용

지문 데이터 속에 사용자가 원하는 데이터를 포함 시킬 수 있는데 이때 지문 데이터 속에 포함되는 사용자 데이터를 Payload 라고 한다. NBioBSP COM 모듈에서는 바이너리 혹은 텍스트 인코딩된 데이터를 이용 할 수 있다.

2.4.1 지문 데이터에 Payload 삽입

Payload 를 지문 데이터에 삽입하는 방법은 Enroll 메소드를 이용하여 지문 데이터를 작성할 때 삽입하는 방법과 이미 작성된 지문 데이터에 CreateTemplate 메소드를 이용하여 삽입하는 방법이 있다.

Enroll 메소드를 이용하는 방법은 삽입을 원하는 Payload 데이터를 Enroll 메소드를 호출할 때 파라미터로 넘겨 주어 Payload 가 삽입된 지문 데이터를 얻을 수 있다.

```
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 ...
```

이미 등록된 지문 데이터에 Payload 데이터를 삽입하는 방법은 CreateTemplate 메소드를 이용하면 된다. CreateTemplate 메소드는 이외에도 기존의 지문 데이터와 신규 지문 데이터를 합치는 역할도 수행할 수 있다.

```
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 ...
```

2.4.2 지문 데이터로부터 Payload 추출

지문 템플릿(등록용 데이터)에 저장되어 있는 Payload 데이터는 Verify 또는 VerifyMatch 메소드를 이용하여 매칭한 결과가 참일 때만 꺼내올 수 있다.

```
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
```

VerifyMatch 메소드를 이용하여 Payload 데이터를 추출하는 방법도 Verify 메소드를 이용하는 방법과 동일하다.

2.5 UI 변경

NBioBSP 에서 기본적으로 제공되는 UI 가 아닌 Customize 된 UI 를 사용하고 싶은 경우 NBioBSP Class Library 모듈에서는 Customized 된 UI 가 들어가 있는 리소스 파일을 읽어올 수 있는 방법을 제공한다.

NBioBSP 는 기본적으로 영문 UI 가 제공되므로 영문 UI 가 아닌 다른 언어의 UI 가 들어있는 리소스 파일을 로드하고 싶은 경우 SetSkinResource 메소드를 사용하면 된다.

```
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!";
     }
}
```

이 때 리소스 파일 경로는 전체 경로를 넘겨주어야 한다. Customized 된 UI 를 작성하는 방법은 별도의 문서로 제공된다.

부록 A. Class Library for .NET Reference

A.1 NBioAPI Class

NBioBSP Class Library 를 사용하기 위한 기본 클래스. 대부분의 기능이 이 클래스에서 지원되며 다수의 다른 클래스를 포함하고 있다. 이 오브젝트는 반드시 선언되어야 합니다.

A.1.1 Basic Method

GetVersion

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

Description
이 함수는 현재 BSP 모듈 버전을 돌려준다.

Parameters
Version:
버전 정보

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨

GetInitInfo

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

Description
초기화 정보를 얻는데 사용한다.

Parameters
InitInfo:
초기화 정보를 담아오기 위한 구조체 포인터.

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨

SetInitInfo

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

Description
초기화 정보 구조체를 설정한다. 현재 NBioAPI 는 type 0 구조체를 지원한다. 이 함수를 호출하기 전에 반드시 InitInfo 구조체를 초기화해야 한다.

Parameters
InitInfo:
초기화 정보를 설정하기 위한 구조체 포인터.

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INIT_MAXFINGER : MaxFingersForEnroll 값이 범위를 벗어남.
NBioAPI.Error.INIT_SAMPLESPERFINGER : 손가락별 샘플 수의 범위를 벗어남.
NBioAPI.Error.INIT_ENROLLQUALITY : Enroll 품질의 범위를 벗어남.
NBioAPI.Error.INIT_VERIFYQUALITY : Verify 품질의 범위를 벗어남.
NBioAPI.Error.INIT_IDENTIFYQUALITY : Identify 품질의 범위를 벗어남.
NBioAPI.Error.INIT_SECURITYLEVEL : 보안 수준의 범위를 벗어남.

EnumerateDevice

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

Description
시스템에 장착된 Device 의 수와 ID 를 얻는다.

Parameters
NumDevice:
장착된 Device 수

DeviceID:
Device ID 리스트를 가지고 있는 배열

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨

GetDeviceInfo

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

Description
지정한 Device 에 대한 정보를 얻는다.

Parameters
DeviceID:
정보를 얻고자 하는 Device ID

DeviceInfo:
Device 정보에 대한 구조체

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.DEVICE_NOT_OPENED: 디바이스가 Open 되지 않았음
NBioAPI.Error.WRONG_DEVICE_ID : 잘못된 디바이스 ID

SetDeviceInfo

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

Description
현재 장착된 Device 에 특정한 설정을 한다. 이미지 폭과 높이는 설정할 수 없다.

Parameters
DeviceID:
설정 하고자 하는 Device ID

DeviceInfo:
Device 정보에 대한 구조체

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.DEVICE_NOT_OPENED : 디바이스가 Open 되지 않았음
NBioAPI.Error.WRONG_DEVICE_ID : 잘못된 디바이스 ID
NBioAPI.Error.DEVICE_BRIGHTNESS : 밝기 값의 범위를 벗어남
NBioAPI.Error.DEVICE_CONTRAST : 고대비값의 범위를 벗어남
NBioAPI.Error.DEVICE_GAIN : Gain 값의 범위를 벗어남

OpenDevice

public System.UInt32 OpenDevice (System.Int16 DeviceID)

Description
NBioBSP 가 사용하기 위해 특정한 Device 를 초기화한다. 만약 Device ID 가 0 이면 기본 Device 를 사용한다. 만약 이 함수를 호출하지 않으면 NBioBSP 의 Capture 나 Enroll 과 같은 함수를 사용할 수 없다.

Parameters
nDeviceID:
Open 하고자 하는 Device ID

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.DEVICE_ALREADY_OPENED : 이미 열려져 있는 디바이스를 다시 초기화 했음
NBioAPI.Error.DEVICE_OPEN_FAIL : 디바이스 초기화 실패

CloseDevice

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

Description
OpenDevice 에 의해 열려진 Device 를 닫고 사용을 중지한다. 반드시 OpenDevice 에 의해 열려진 DeviceID 와 동일한 값을 파라메터로 넘겨야 한다.

Parameters
DeviceID:
사용중지 하고자 하는 Device ID

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.DEVICE_NOT_OPENED : 디바이스가 Open 되지 않았음
NBioAPI.Error.WRONG_DEVICE_ID : 잘못된 디바이스 ID

AdjustDevice

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

Description
현재 Open 된 Device 의 밝기조절을 할 수 있는 Dialog 를 띄운다.

Parameters
WindowOption:
NBioBSP 의 윈도우 속성

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.DEVICE_NOT_OPENED : 디바이스가 Open 되지 않았음
NBioAPI.Error.WRONG_DEVICE_ID : 잘못된 디바이스 ID
NBioAPI.Error.USER_CANCEL : 밝기 조절 중 사용자가 취소 버튼을 누르고 취소함

GetOpenedDeviceID

public System.Int16 GetOpenedDeviceID ( );

Description
연결된 Device 중에서 열려 있는 Device 의 ID 를 얻는다.

Parameters
N/A

Return Value
현재 열려진 Device ID 값을 리턴한다.

CheckFinger

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

Description
현재 Open 된 디바이스의 Sensor 에 손가락을 올려놓았는지를 검사한다. 이 함수는 현재 USB 디바이스에서만 지원한다.
HFDU 01/04/06 의 경우 Device Driver v4.1.0.1 이상, HFDU11/14 는 모든 버전, HFDU 05/07 은 지원하지 않는다.

Parameters
bExistFinger:
손가락 유무, 만약 손가락이 Sensor 위에 있으면 NBioAPI_TRUE 를, 그렇지 않으면 NBioAPI_FALSE 값을 가진다.

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.DEVICE_NOT_OPENED : 디바이스가 Open 되지 않았음
NBioAPI.Error.LOWVERSION_DRIVER : 디바이스 드라이버의 버전이 낮아 지원되지 않음.

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

NBioBSP 모듈에 있는 FIRHandle 로부터 FIR 을 얻는다. 어플리케이션에서 NBioAPI_FIR 구조체 버퍼를 할당해야만 한다. 오버로드된 두개의 매소드가 있다.

Parameters

hFIR:

NBioBSP 모듈에 있는 FIR handle

FIR:

지문정보가 담겨진 FIR 클래스

Format:

FIR 의 Format. Format 은 현재 NBioAPI_FIR_FORMAT_STANDARD 만 지원 한다.

Return Value

NBioAPI.Error.NONE : 정상적으로 수행됨

NBioAPI.Error.INVALID_HANDLE : handle 값이 유효하지 않음

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

NBioBSP 모듈에 있는 FIRHandle 로부터 FIR header 정보를 얻는다. 오버로드된 두개의 매소드가 있다.

Parameters

hFIR:

NBioBSP 모듈에 있는 FIR handle

Header:

FIR header

Format:

FIR 의 Format. Format 은 현재 NBioAPI_FIR_FORMAT_STANDARD 만 지원 한다.

Return Value

NBioAPI.Error.NONE : 정상적으로 수행됨

NBioAPI.Error.INVALID_HANDLE : handle 값이 유효하지 않음

GetTextFIRFromHandle

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

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

Description

NBioBSP 모듈에 있는 FIRHandle 로부터 텍스트 인코딩 된 FIR 데이터를 얻는다. 오버로드된 두개의 매소드가 있다.

Parameters

hFIR:

NBioBSP 모듈의 FIR handle

TextFIR:

텍스트 FIR 구조체의 포인터

bIsWide:

만약 프로그램에서 Unicode char 를 사용한다면 NBioAPI_TRUE 를 설정해야만 한다.

Format:

FIR 의 Format. Format 은 현재 NBioAPI_FIR_FORMAT_STANDARD 만 지원 한다.

Return Value

NBioAPI.Error.NONE : 정상적으로 수행됨

NBioAPI.Error.INVALID_HANDLE : handle 값이 유효하지 않음

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

이 매소드는 지정한 용도에 맞게 Capture 를 하고 FIR 핸들을 리턴한다. FIR 의 용도는 CapturedFIR 의 header 에 기록된다. 만약 Audit data 가 NULL 이 아니면 “Raw”타입의 FIR 을 리턴한다. 오버로드된 3 개의 메소드가 있다.

Parameters

CapturedFIR:

Capture 한 데이터를 포함하는 FIR 의 handle. 이 데이터는 FIR 의 “Intrmediate” 형태이거나(용도에 따라 Process 나 CreateTemplate() 함수들에 사용된다) “Processed” 형태를 가진다(용도에 따라 VerifyMatch 에 직접 사용된다).

Purpose:

Capture 의 용도. 이 값에 따라 등록이나 인증 하는 UI 가 선택적으로 display 된다.

Timeout:

작동을 위한 timeout 값(millisecond 단위)을 정수형으로 지정한다. 만약 timeout 이 되면 함수는 결과값 없이 error 를 리턴한다. 이 값은 임의의 양수가 될 수 있다. 만약 값이 NBioAPI.Type.TIMEOUT.DEFAULT 이라면 기본값을 사용할 것이다. 만약 값이 NBioAPI.Type.TIMEOUT.INFINITE 이라면 timeout 이 없다.

AuditData:

원시 지문이미지 데이터를 가지고 있는 FIR handle. 이 데이터는 장치에서 사람의 인체인지 확인 할 수 있는 데이터를 제공한다. 만약 입력할 때 null 이면 audit data 는 수집하지 않는다.

WindowOption:

NBioBSP 의 윈도우 속성, 입력 값이 null 이면 기본값을 사용한다.

Return Value

NBioAPI.Error.NONE : 정상적으로 수행됨

NBioAPI.Error.DEVICE_NOT_OPENED : 디바이스가 Open 되지 않았음

NBioAPI.Error.USER_CANCEL : Capture 중 사용자가 취소 버튼을 누르고 취소함

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_TEXTENCODE 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

이 함수는 인증이나 인식의 용도를 위해 Capture 메소드를 호출해서 얻은 중간 데이터를 처리한다. 만약 현재 모듈에서 처리할 수 있는 능력이 있다면 “Processed” FIR 을 만들고 그렇지 않다면 ProcessedFIR 은 NULL 로 설정될 것이다. NBioBSP 는 Capture 나 Enroll 시 내부적으로 Processed Data 를 생성하므로 이 메소드를 다시 불러줄 필요는 없다. 단지 Audit 데이터로 받은 “Raw” 데이터를 Processing 하는데 사용되어 질 수 있다.

Parameters

CapturedFIR:

Capture 한 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다.

ProcessedFIR:

새롭게 구성된 “Processed”FIR

Return Value

NBioAPI.Error.NONE : 정상적으로 수행됨

NBioAPI.Error.INVALID_HANDLE : 핸들값이 유효하지 않음

NBioAPI.Error.ALREADY_PROCESSED : 이미 “Processed”된 FIR 임

NBioAPI.Error.DATA_PROCESS_FAIL : Process 를 할 수 없는 Data 임

NBioAPI.Error.ENCRYPTED_DATA_ERROR : 암호화된 데이터를 해독 할 수 없음

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL : 데이터가 변조되었음

CreateTemplate

```
public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

public System.UInt32 CreateTemplate ( NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR CapturedFIR ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.INPUT_FIR StoredTemplate ,
                                      out NITGEN.SDK.NBioBSP.NBioAPI.Type.HFIR NewTemplate ,
                                      NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );
```

Descriptions

이 함수는 새로운 등록 Template 를 생성 하기 위한 용도로 원시 지문 데이터를 갖는 FIR 을 갖는다. 새로운 FIR 은 CapturedFIR 로부터 구성되고, 현재 존재하는 StoredTemplate 를 기준으로 개조를 수행한다. 만약 StoredTemplate 가 Payload 를 가지고 있다 하더라도 Payload 는 NewTemplate 에 복사 되지 않고 전에 있던 StoredTemplate 는 변하지 않는다. StoredTemplate 가 NULL 일 경우 CapturedFIR 과 Payload 만을 가지고 NewTemplate 를 생성한다. 만약 NewTemplate 가 Payload 를 필요로 한다면 메소드에 인수로 넘겨야 한다.

Parameters

CapturedFIR:

Capture 된 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다. 오직 'Verification 을 위한 Enroll' 의 용도로 설정된 것만을 CapturedFIR 에 받아 들인다.

StoredTemplate:

선택적으로 Template 를 개조할 수 있다.

NewTemplate:

CapturedFIR 과 (선택적으로)StoredTemplate 로부터 파생되어진 새롭게 생성된 Template 의 핸들

Payload:

새롭게 생성된 Tmeplate 안에 삽입될 데이터. 만약 null 이면 이 인수는 무시 된다.

Return Value

NBioAPI.Error.NONE : 정상적으로 수행됨

NBioAPI.Error.INVALID_HANDLE : 핸들값이 유효하지 않음

NBioAPI.Error.ENCRYPTED_DATA_ERROR : 암호화된 데이터를 해독 할 수 없음

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL : 데이터가 변조되었음

NBioAPI.Error.MUST_BE_PROCESSED_DATA : "Processed" 데이터가 아님

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_TEXTENCODE 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_TEXTENCODE StoredTemplate ,
                                   out System.Boolean Result ,
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_PAYLOAD Payload );

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 );

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 );

public System.UInt32 VerifyMatch ( NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE CapturedFIR ,
                                   NITGEN.SDK.NBioBSP.NBioAPI.Type.FIR_TEXTENCODE 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

이 함수는 두개의 FIR(CapturedFIR 과 StoredTemplate) 간에 인증 매칭(1-to-1)을 수행한다. CapturedFIR 은 이 인증을 위해 특별히 구성된 “Prosessed”FIR 이다. StoredTemplate 는 등록 시에 만들어진 것으로, 만약 StoredTemplate 가 Payload 를 가지고 있다면 성공적으로 인증하고 나서 그 Payload 를 리턴한다.

Parameters

CapturedFIR:

인증하고자 하는 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다.

StoredTemplate:

인증에 대비한 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다.

Result:

FIR 에 의해 정합이 됐는지 안됐는지 지시하는 Boolean 값(NBioAPI.Type.TRUE / NBioAPI.Type.FALSE) 을 가진다.

Payload:

만약 StoredTemplate 가 Payload 를 가지고 있고, Result 값이 TRUE 라면 NBioAPI.Type.FIR_PAYLOAD 구조체에 할당된 Payload 를 리턴한다.

Return Value

NBioAPI.Error.NONE : 정상적으로 수행됨

NBioAPI.Error.INVALID_HANDLE : 핸들값이 유효하지 않음

NBioAPI.Error.ENCRYPTED_DATA_ERROR : 암호화된 데이터를 해독 할 수 없음

NBioAPI.Error.INTERNAL_CHECKSUM_FAIL : 데이터가 변조되었음

NBioAPI.Error.MUST_BE_PROCESSED_DATA: “Processed” 데이터가 아님

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_TEXTENCODE 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.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

이 함수는 두개의 FIR(CapturedFIR 과 StoredTemplate) 간에 인증 매칭(1-to-1)을 수행한다. CapturedFIR 은 이 인증을 위해 특별히 구성된 “Prosessed”FIR 이다. StoredTemplate 는 등록 시에 만들어진 것으로, 만약 StoredTemplate 가 Payload 를 가지고 있다면 성공적으로 인증하고 나서 그 Payload 를 리턴한다.
매칭을 위해 NBioAPI_MATCH_OPTION 을 주고 매칭을 수행할 수 있는 것이 NBioAPI_VerifyMatch() 함수와 유일하게 다른점이다.

Parameters

CapturedFIR:
인증하고자 하는 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다.

StoredTemplate:
인증에 대비한 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다.

Result:
FIR 에 의해 정합이 됐는지 안됐는지 지시하는 Boolean 값(NBioAPI.Type.TRUE / NBioAPI.Type.FALSE) 을 가진다.

Payload:
만약 StoredTemplate 가 Payload 를 가지고 있고, Result 값이 TRUE 라면 NBioAPI.Type.FIR_PAYLOAD 구조체에 할당된 Payload 를 리턴한다.

MatchOption:
NBioAPI.Type.MATCH_OPTION_0 구조체를 통해 매칭을 위한 조건을 지정해 줄 수 있다.

Return Value

NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INVALID_HANDLE : 핸들값이 유효하지 않음
NBioAPI.Error.ENCRYPTED_DATA_ERROR : 암호화된 데이터를 해독 할 수 없음
NBioAPI.Error.INTERNAL_CHECKSUM_FAIL : 데이터가 변조되었음
NBioAPI.Error.MUST_BE_PROCESSED_DATA: “Processed” 데이터가 아님

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_TEXTENCODE 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

이 함수는 장착된 Device 로부터 지문 데이터를 Capture 해서 등록용으로 ProcessedFIR 을 생성한다.

Parameters

StoredTemplate:

선택적으로 적용되는 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다. Template 를 개조할 수 있다.

NewTemplate:

CapturedFIR 과 StoredTemplate(선택적으로)로부터 파생 되어진 새롭게 생성된 Template 의 핸들.

Payload:

새롭게 생성된 Template 에 포함될 Data

Timeout:

작동을 위한 timeout 값(millisecond 단위)을 정수형으로 지정한다. 만약 timeout 이 되면 함수는 결과값 없이 error 를 리턴한다. 이 값은 임의의 양수가 될 수 있으며 만약 값이 NBioAPI.Type.TIMEOUT.DEFAULT 이라면 기본값을 사용할 것이다.

AuditData:

원시 지문이미지 데이터를 가지고 있는 FIR handle. 이 데이터는 장치에서 사람의 인체인지 확인할 수 있는 데이터를 제공한다.
만약 입력할 때 null 값이면 audit data 를 수집하지 않는다.
WindowOption:
NBioBSP 의 윈도우 속성. 입력 값이 null 이면 기본값을 사용한다.

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INVALID_HANDLE : 핸들값이 유효하지 않음
NBioAPI.Error.DEVICE_NOT_OPENED: 디바이스가 Open 되지 않았음
NBioAPI.Error.ENCRYPTED_DATA_ERROR : 암호화된 데이터를 해독 할 수 없음
NBioAPI.Error.INTERNAL_CHECKSUM_FAIL : 데이터가 변조되었음
NBioAPI.Error.FUNCTION_FAIL : 변환 실패
NBioAPI.Error.USER_CANCEL : Enroll 중 사용자가 취소 버튼을 누르고 취소함

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

이 함수는 장착된 Device 로부터 지문 데이터를 Capture 해서 StoredTemplate 와 비교한다.

Parameters

StoredTemplate:

인증 시 대조하기 위한 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다.

Result :
정합 결과

Payload:
만약 StoredTemplate 가 Payload 를 가지고 있다면 할당된 Data 구조에 리턴한다.

Timeout:
작동을 위한 timeout 값(millisecond 단위)을 정수형으로 지정한다. 만약 timeout 이 되면 함수는 결과값 없이 error 를 리턴한다.
이 값은 임의의 양수로 표현되며 만약 값이 NBioAPI.Type.TIMEOUT.DEFAULT 이라면 기본값을 사용할 것이다.

AuditData:
원시 지문이미지 데이터를 가지고 있는 FIR handle. 이 데이터는 장치에서 사람의 인체인지 확인할 수 있는 데이터를 제공한다.
만약 입력할 때 null 이면 audit data 를 수집하지 않는다.

WindowOption:
NBioBSP 의 윈도우 속성. 입력한 값이 null 이면 기본값을 사용한다.

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INVALID_HANDLE : 핸들값이 유효하지 않음
NBioAPI.Error.USER_CANCEL : Verify 중 사용자가 취소 버튼을 누르고 취소함
NBioAPI.Error.ENCRYPTED_DATA_ERROR : 암호화된 데이터를 해독 할 수 없음
NBioAPI.Error.INTERNAL_CHECKSUM_FAIL : 데이터가 변조되었음

A.1.4 User Interface Method

SetSkinResource

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

Descriptions

BSP 모듈에 새로운 스킨 리소스를 설정한다. 스킨 리소스는 OEM 사용자를 위해 만들어질 수 있다.

Parameters

szResPath:

리소스 DLL 의 경로. 만약 이 경로가 NULL 로 되어 있다면 BSP 는 기본 리소스를 사용한다.

Return Value

NBioAPI.Type.TRUE : 함수의 성공적인 작업 완료

NBioAPI.Type.FALSE : 리소스 DLL 을 찾지 못했음. 이런 경우 BSP 는 기본 리소스를 사용한다.

A.2 NBioAPI.Export Class

데이터 변환 관련 기능이 들어 있는 클래스

Export

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

Descriptions
클래스 생성자. 초기값으로 NBioAPI 클래스를 넘겨주어야 한다.

Parameters
NBioBSP:
NBioAPI 클래스

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
FDx Data(400byte minutiae array) 를 FIR 형태로 변환한다. 변환한 데이터는 FIR handle 이 된다. 이 함수에 Raw data 형태는 받아 들이지 않는다.

Parameters
FDxData:
FDx 데이터 배열

FDxDataType :
FDx data 타입들이다.

Purpose:
ProcessedFIR 의 용도

ProcessedFIR:
처리된 데이터를 가지고 있는 FIR 의 handle

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INVALID_MINSIZE : minutiae 크기가 유효하지 않다.
NBioAPI.Error.FUNCTION_FAIL : 변환 실패

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
FDx Data 를 FIR 형태로 변환한다. 변환한 데이터는 FIR handle 이 된다. 이 함수에 Raw data 형태는 받아 들이지 않는다.

Parameters
FDxData:
FDx 데이터 배열

FDxTemplateSize:
Template 하나의 크기값.

FDxDataType :
FDx data 타입들이다.

Purpose:
ProcessedFIR 의 용도

ProcessedFIR:
처리된 데이터를 가지고 있는 FIR 의 handle

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INVALID_MINSIZE : minutiae 크기가 유효하지 않다.
NBioAPI.Error.FUNCTION_FAIL : 변환 실패

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
FIR 데이터를 FDx 데이터로 변환한다.

Parameters
CapturedFIR:
변환될 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다.

ExportData:
NBioAPI.Export.EXPORT_DATA 구조체. 이 구조체는 FIR 과 FDx 데이터에 대한 모든 정보를 가지고 있다.

FDxDataType :
FDx 데이터 타입

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.MUST_BE_PROCESSED_DATA : “Processed” 데이터가 아님
NBioAPI.Error.UNKNOWN_FORMAT : 알 수 없는 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
FIR 데이터를 Image 데이터로 변환한다.

Parameters
AuditFIR:
변환하게 될 FIR 의 핸들이나 바이너리 FIR, 또는 텍스트 인코딩 된 FIR 이 될 수 있다.

ExportAuditData:
NBioAPI.Export.EXPORT_AUDIT_DATA 구조체. 이 구조체는 FIR 과 FDx 데이터의 모든 정보를 가지고 있다.

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.ALREADY_PROCESSED : 이미 “Processed”된 FIR 임

ImageToNBioBSP

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

Descriptions
Raw image data 로부터 FIR 형태로 변환한다.

Parameters
ExportAuditData:
변환하고자 하는 Minutiae 데이터를 가진 구조체

FIR:
변환된 FIR 을 갖는 handle

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INVALID_MINSIZE : minutiae 크기가 유효하지 않음.

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
Minutiae data 로부터 FIR 형태로 변환한다.

Parameters
ExportData:
변환하고자 하는 Minutiae 데이터를 가진 구조체

Purpose:
FIR 의 용도를 지정

ProcessedFIR:
변환된 FIR 을 갖는 handle

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INVALID_MINSIZE : minutiae 크기가 유효하지 않음.

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
Minutiae data 로부터 FIR 형태로 변환한다.

Parameters
ExportData:
변환하고자 하는 Minutiae 데이터를 가진 구조체

Purpose:
FIR 의 용도를 지정

DataType:
Minutiae 데이터의 타입 (RAW, PROCESSED, …)

ProcessedFIR:
변환된 FIR 을 갖는 handle

Return Value
NBioAPI.Error.NONE : 정상적으로 수행됨
NBioAPI.Error.INVALID_MINSIZE : minutiae 크기가 유효하지 않음.

A.3 NBioAPI.IndexSearch Class

IndexSearch 를 이용한 데이터 검색 관련 기능 클래스

A.3.1 Initialization Method

IndexSearch

public System.Uint32 IndexSearch (NITGEN.SDK.NBioBSP.NBioAPI NBioBSP);

Descriptions
클래스 생성자. 초기값으로 NBioAPI 클래스를 넘겨주어야 한다.

Parameters
NBioBSP:
NBioAPI 클래스

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

InitEngine

public System.UInt32 InitEngine ( );

Descriptions
IndexSearch Engine 을 초기화한다. 이 함수는 IndexSearch Engine 의 램상주 DB 에 메모리를 할당하고 Global Variable 을 초기화 한다.

Parameters
없음.

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error. INDEXSEARCH_INIT_FAIL : 초기화 실패.

TerminateEngine

public System.UInt32 TerminateEngine ( );

Descriptions
IndexSearch Engine 을 종료한다. 사용하는 어플리케이션을 종료할 때에는 반드시 이 함수를 호출하여 IndexSearch Engine 에서 사용하고 있던 메모리를 해제한다.

Parameters
없음.

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.

GetInitInfo

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

Descriptions
현재 설정되어있는 IndexSearch Engine 의 Parameter Value 들을 읽어온다. 비어있는 NBioAPI_INDEXSEARCH_INIT_INFO_0 구조체의 포인터를 전달하면 Parameter Value 가 기록되어 출력된다. 현재는 StructType 의 값을 0 만 지원한다.

Parameters
InitInfo0:
사용자 설정 parameter

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.

SetInitInfo

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

Descriptions
Engine 의 parameter 를 설정한다. 현재는 StructType 의 값을 0 만 지원한다.

Parameters
InitInfo0:
사용자 설정 parameter

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.INIT_PRESEARCHRATE : PreSearchRate 의 값이 유효하지 않음.

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

지문 데이터와 사용자 ID 를 입력하여 IndexSearch Engine 의 램상주 지문 DB 에 지문을 등록한다. 등록이 끝나면 DB 에 등록된 지문 Template 에 대한 정보를 얻어온다.

Parameters

CapturedFIR:

FIR 데이터 (자세한 설명은 NBioAPI 매뉴얼 참조)

UserID:

사용자 ID

SampleInfo:

등록된 지문 Template 에 대한 세부적인 정보. (등록 지문의 손가락별 등록된 Sample 의 개수를 나타낸다.)

Return Values

NBioAPI.Error.NONE : 정상적으로 수행됨.

NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.

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
사용자 ID, 손가락 번호, 샘플번호로 이루어진 Template 정보(pFpInfo)를 입력하여 해당 사용자의 지문을 IndexSearch Engine 의 램상주 지문 DB 로부터 삭제한다.

Parameters
FpInfo:
사용자 ID, 손가락 번호, 샘플번호로 이루어진 Template 정보

UserID:
사용자 ID

FingerID:
손가락 번호

SampleNumber :
샘플 번호

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.

RemoveUser

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

Descriptions
사용자 ID 를 입력하여 해당 사용자의 모든 지문 Template 을 IndexSearch Engine 의 램상주 지문 DB 로부터 삭제한다.

Parameters
UserID:
사용자 ID

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.

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

입력 지문이 Fingerprint DB 에 존재하는지 검색하고 그 결과를 얻는다. 지문 데이터와 Security Level(nSecuLevel), 비어있는 NBioAPI_INDEXSEARCH_FP_INFO structure (pFpInfo)를 입력하면 출력으로 동일한 지문이 존재할 경우, pFpInfo structure 에 지문 정보가 출력되고 동일 지문이 없을 경우, 모든 정보가 0 으로 set 되어 출력된다. Security Level 은 입력지문과 비교 지문이 동일한 지문인지를 판단하는 임계값으로서 1 에서 9 까지의 값을 가지고 9 에 가까운 값을 입력할수록 비교지문과의 유사도가 높아야 동일 지문으로 판단한다.
마지막 파라메터로 Callback 함수를 등록 할 경우 내부적으로 매칭 직전마다 등록한 Callback 함수가 호출되어 매칭 여부를 제어하거나 매칭을 중지 할 수 있다. 자세한 설명은 NBioAPI_INDEXSEARCH_CALLBACK_INFO_0 structure 를 참조하면 된다.

Parameters
CapturedFIR:
FIR 데이터 (자세한 설명은 NBioAPI 매뉴얼 참조)

SecuLevel:
Security level (1~9) 입력지문과 비교 지문이 동일한 지문인지를 판단하는 임계 값

FpInfo:
Identification 결과 지문 정보

CallbackInfo:
Search 중에 호출될 Callback 함수에 대한 정보를 넘길 수 있다. 이 값을 NULL 로 하면 Callback 함수는 불리지 않는다.

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.INDEXSEARCH_IDENTIFY_STOP : 지문 검색 중 Callback 함수에 의해 검색이 중단되었음.

A.3.2 DB Method

SaveDBToFile

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

Descriptions
IndexSearch Engine 의 램상주 지문 DB 를 Disk 에 저장하여 Backup 한다. 저장될 파일의 폴더를 포함한 Full-Path file name 을 입력하여 이 함수를 호출하면 지정된 위치에 램상주 DB 의 내용이 파일로 저장된다.

Parameters
szFilepath:
저장될 DB 파일의 폴더를 포함한 Full-Path file name

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.INDEXSEARCH_SAVE_DB : 지문 DB 저장 오류

LoadDBFromFile

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

Descriptions
Disk 에 Backup 된 IndexSearch Engine DB 를 Disk 로부터 읽어 IndexSearch Engine 의 램상주 지문 DB 에 로드한다.

Parameters
szFilepath:
로드 할 DB 파일의 디렉터리를 포함한 Full-Path file name

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.INDEXSEARCH_LOAD_DB : Backup 지문 DB 로드 오류

ClearDB

public System.UInt32 ClearDB ( );

Descriptions
IndexSearch Engine 의 램상주 지문 DB 에 있는 모든 지문 데이터를 삭제한다.

Parameters
없음.

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.

GetDataCount

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

Descriptions
IndexSearch Engine 의 램상주 지문 DB 에 있는 지문 데이터의 개수를 얻어온다.

Parameters
DataCount:
지문 DB 의 개수

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.

CheckDataExist

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

Descriptions
IndexSearch Engine 의 램상주 지문 DB 에 특정 지문 데이터가 있는지 검사한다.

Parameters
FpInfo:
사용자 ID, 손가락 번호, 샘플번호로 이루어진 Template 정보

Exist:
존재 유무 값

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.INDEXSEARCH_INIT_FAIL : 초기화 되지 않았음.

A.4 NBioAPI.NSearch Class

NSearch 를 이용한 데이터 검색 관련 기능 클래스

A.4.1 Initialization Method

NSearch

public System.Uint32 NSearch (NITGEN.SDK.NBioBSP.NBioAPI NBioBSP);

Descriptions
클래스 생성자. 초기값으로 NBioAPI 클래스를 넘겨주어야 한다.

Parameters
NBioBSP:
NBioAPI 클래스

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

InitEngine

public System.UInt32 InitEngine ( );

Descriptions
NSearch Engine 을 초기화한다. 이 함수는 NSearch Engine 의 램상주 DB 에 메모리를 할당하고 Global Variable 을 초기화 한다.

Parameters
없음.

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_OPEN_FAIL : Engine 열기 실패.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 실패

TerminateEngine

public System.UInt32 TerminateEngine ( );

Descriptions
NSearch Engine 을 종료한다. 사용하는 어플리케이션을 종료할 때에는 반드시 이 함수를 호출하여 NSearch Engine 에서 사용하고 있던 메모리를 해제한다.

Parameters
없음.

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

GetInitInfo

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

Descriptions
현재 설정되어있는 NSearch Engine 의 Parameter Value 들을 읽어온다. 비어있는 NBioAPI_NSEARCH_INIT_INFO_0 구조체의 포인터를 전달하면 Parameter Value 가 기록되어 출력된다. 현재는 StructType 의 값을 0 만 지원한다.

Parameters
InitInfo0:
사용자 설정 parameter

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

SetInitInfo

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

Descriptions
Engine 의 parameter 를 설정한다. 현재는 StructType 의 값을 0 만 지원한다.

Parameters
InitInfo0:
사용자 설정 parameter

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.INIT_MAXCANDIDATE : MaxCandidate 값이 유효하지 않음.

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_TEXTENCODE 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

지문 데이터와 사용자 ID 를 입력하여 NSearch Engine 의 램상주 지문 DB 에 지문을 등록한다. 등록이 끝나면 DB 에 등록된 지문 Template 에 대한 정보를 얻어온다.

Parameters

CapturedFIR:

FIR 데이터 (자세한 설명은 NBioAPI 매뉴얼 참조)

UserID:

사용자 ID

SampleInfo:

등록된 지문 Template 에 대한 세부적인 정보. (등록 지문의 손가락별 등록된 Sample 의 개수를 나타낸다.)

Return Values

NBioAPI.Error.NONE : 정상적으로 수행됨.

NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

NBioAPI.Error.NSEARCH_OVER_LIMIT : 템플릿 제한 개수 초과.

NBioAPI.Error.NSEARCH_MEM_OVERFLOW : 메모리 할당 실패.

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
사용자 ID, 손가락 번호, 샘플번호로 이루어진 Template 정보(pFpInfo)를 입력하여 해당 사용자의 지문을 NSearch Engine 의 램상주 지문 DB 로부터 삭제한다.

Parameters
FpInfo:
사용자 ID, 손가락 번호, 샘플번호로 이루어진 Template 정보

UserID:
사용자 ID

FingerID:
손가락 번호

SampleNumber:
샘플 번호

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

RemoveUser

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

Descriptions
사용자 ID 를 입력하여 해당 사용자의 모든 지문 Template 을 NSearch Engine 의 램상주 지문 DB 로부터 삭제한다.

Parameters
UserID:
사용자 ID

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

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_TEXTENCODE 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

입력 지문이 Fingerprint DB 에 존재하는지 검색하고 그 결과를 Candidate 배열로 반환한다.

Parameters

CapturedFIR:

FIR 데이터 (자세한 설명은 NBioAPI 매뉴얼 참조)

Candidate:

SearchData 결과 리스트

Return Values

NBioAPI.Error.NONE : 정상적으로 수행됨.

NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

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
입력 지문이 Fingerprint DB 에 존재하는지 검색하고 그 결과를 얻는다. 지문 데이터와 Security Level(nSecuLevel), 비어있는 NBioAPI_NSEARCH_FP_INFO structure (pFpInfo)를 입력하면 출력으로 동일한 지문이 존재할 경우, pFpInfo structure 에 지문 정보가 출력되고 동일 지문이 없을 경우, 모든 정보가 0 으로 set 되어 출력된다. Security Level 은 입력지문과 비교 지문이 동일한 지문인지를 판단하는 임계값으로서 1 에서 9 까지의 값을 가지고 9 에 가까운 값을 입력할수록 비교지문과의 유사도가 높아야 동일 지문으로 판단한다.

Parameters
CapturedFIR:
FIR 데이터 (자세한 설명은 NBioAPI 매뉴얼 참조)

SecuLevel:
Security level (1~9) 입력지문과 비교 지문이 동일한 지문인지를 판단하는 임계 값

FpInfo:
Identification 결과 지문 정보

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.NSEARCH_IDENTIFY_FAIL : 입력 지문과 일치하는 지문이 존재하지 않음.

A.4.2 DB Method

SaveDBToFile

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

Descriptions
NSearch Engine 의 램상주 지문 DB 를 Disk 에 저장하여 Backup 한다. 저장될 파일의 폴더를 포함한 Full-Path file name 을 입력하여 이 함수를 호출하면 지정된 위치에 램상주 DB 의 내용이 파일로 저장된다.

Parameters
szFilepath:
저장될 DB 파일의 폴더를 포함한 Full-Path file name

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.NSEARCH_SAVE_DB : 지문 DB 저장 오류

LoadDBFromFile

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

Descriptions
Disk 에 Backup 된 NSearch Engine DB 를 Disk 로부터 읽어 NSearch Engine 의 램상주 지문 DB 에 로드한다.

Parameters
szFilepath:
로드 할 DB 파일의 디렉터리를 포함한 Full-Path file name

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.NSEARCH_LOAD_DB : Backup 지문 DB 로드 오류

ClearDB

public System.UInt32 ClearDB ( );

Descriptions
NSearch Engine 의 램상주 지문 DB 에 있는 모든 지문 데이터를 삭제한다.

Parameters
없음.

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

GetDataCount

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

Descriptions
NSearch Engine 의 램상주 지문 DB 에 있는 지문 데이터의 개수를 얻어온다.

Parameters
DataCount:
지문 DB 의 개수

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

CheckDataExist

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

Descriptions
NSearch Engine 의 램상주 지문 DB 에 특정 지문 데이터가 있는지 검사한다.

Parameters
FpInfo:
사용자 ID, 손가락 번호, 샘플번호로 이루어진 Template 정보

Exist:
존재 유무 값

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.

ImportIndexSearchDB

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

Descriptions
Disk 에 Backup 된 IndexSearch Engine DB 를 Disk 로부터 읽어 NSearch Engine 의 램상주 지문 DB 에 로드한다.

Parameters
szFilepath:
로드 할 DB 파일의 디렉터리를 포함한 Full-Path file name

Return Values
NBioAPI.Error.NONE : 정상적으로 수행됨.
NBioAPI.Error.NSEARCH_INIT_FAIL : 초기화 되지 않았음.
NBioAPI.Error.NSEARCH_LOAD_DB : Backup 지문 DB 로드 오류