

**NITGEN®**

**eNSearch  
Identification  
Software**

**NITGEN®**

**eNSearch  
Identification  
Software**

**NITGEN®**

**eNSearch  
Identification  
Software**

**NITGEN®**

**eNSearch  
Identification  
Software**

**NITGEN®**

**eNSearch**

**Engine**

**Programmer's**

**Guide**



## **INDEX**

<b>1. INTRODUCTION.....</b>	<b>6</b>
1.1 FEATURES .....	7
1.2 FUNCTIONS .....	8
<b>2. INSTALLATION .....</b>	<b>11</b>
2.1 SYSTEM REQUIREMENT .....	11
2.2 INSTALLATION .....	12
2.3 DIRECTORIES & FILES .....	14
2.4 DEMO PROGRAM.....	16
2.4.1 UI DESCRIPTION .....	16
2.4.2 FINGERPRINT REGISTRATION .....	16
2.4.3 DB MANAGEMENT .....	17
2.4.4 SEARCH/IDENTIFICATION .....	18
2.5 LICENSE SETUP .....	20
<b>3. SDK PROGRAMMING .....</b>	<b>22</b>
3.1 NSEARCH ENGINE STRUCTURE.....	22
3.2 SYSTEM STRUCTURE .....	24
3.3 DB STRUCTURE .....	28
3.4 NSEARCH ENGINE APIs.....	31
3.4.1 INITIALIZATION/TERMINATION/PARAMETER SETTING .....	32
3.4.2 FINGERPRINT REGISTRATION / DELETION / SEARCH .....	35
3.4.3 MRFDB MANAGEMENT .....	41
3.5 VISUAL BASIC PROGRAMMING .....	43
3.5.1 INITIALIZATION / TERMINATION / PARAMETER SETTING .....	44
3.5.2 FINGERPRINT REGISTRATION / DELETION / SEARCH .....	46
3.5.3 MRFDB MANAGEMENT .....	51
3.6 DELPHI PROGRAMMING.....	54
3.6.1 INITIALIZATION / TERMINATION / PARAMETER SETTING .....	55
3.6.2 FINGERPRINT REGISTRATION / DELETION / SEARCH .....	57

3.6.3 MRFDB MANAGEMENT .....	62
<b>3.7 C# (.NET) PROGRAMMING .....</b>	<b>64</b>
3.7.1 INITIALIZATION / TERMINATION / PARAMETER SETTING .....	65
3.7.2 FINGERPRINT REGISTRATION / DELETION / SEARCH .....	67
3.7.3 MRFDB MANAGEMENT .....	72
 <b><u>APPENDIX A. API REFERENCE .....</u></b>	<b><u>74</u></b>
 A.1 FUNCTIONS .....	75
A.2 NSEARCH ENGINE STRUCTURE .....	91
A.3 ERROR CODES & CONSTANTS .....	94
 <b><u>APPENDIX B. COM REFERENCE .....</u></b>	<b><u>96</u></b>
 B.1 PROPERTIES ( SEARCH OBJECT ) .....	96
B.2 PROPERTIES ( CANDIDATELIST OBJECT ) .....	97
B.3 METHODS.....	97
 <b><u>APPENDIX C. .NET REFERENCE .....</u></b>	<b><u>102</u></b>
 C.1 NSEARCH ENGINE METHODS .....	102
C.2 NSEARCH ENGINE STRUCTURE .....	119

# 1



# Introduction

1.1 Features

1.2 Functions

# 1. Introduction

eNSearch SDK(or NSearch SDK) is the 1:N matching DK to enable programmers to develop a fingerprint identification system with large volume of database in high-speed matching performance.

NSearch SDK is composed of two different types of application as follows.

- When you do not know the ID:  
Missing Child Searching System, Criminal Investigation, etc.
- When you do not input the ID for more convenience:  
Access Controller, Membership Management, etc.

The NSearch SDK is designed for these applicable cases to build the 1:N matching system with high performance and accuracy.

## 1.1 Features

### **Optimized API**

NSearch SDK provides the optimized APIs for fingerprint enrollment and identification and helps programmers to integrate a fingerprint identification system in short period.

### **Fast Fingerprint Searching Speed**

NSearch Engine uses a new matching algorithm using indexing techniques, not using serial matching method, and it can search a matched person in a large database as fast as 10,000 fingerprints per second.

### **High Accuracy**

The NSearch Engine can provide more accurate matching results than any other 1:N fingerprint matching algorithm, especially in large volume of database.

### **High Compatibility**

The NSearch Engine, programmed by the ANSI C code, can be easily integrated to any system. The NSearch Engine supports Windows NT/2000/XP.

## 1.2 Functions

The NSearch Engine must be used to enroll and match fingerprint template, by using the NITGEN's NBioBSP module. The followings are simple description about the functions used in the NSearch Engine.

### 1) Initialization / Termination / Parameter setting

#### **NBioAPI\_InitNSearchEngine**

Initializes the Memory Resident Fingerprint DB (MRFDB) with memory allocation and global variable settings.

#### **NBioAPI\_TerminateNSearchEngine**

Deletes the MRFDB in the memory allocated.

#### **NBioAPI\_SetNSearchInitInfo**

Configures the basic parameters used in the NSearch Engine.

#### **NBioAPI\_GetNSearchInitInfo**

Reads the basic parameters used in the NSearch Engine.

### 2) Fingerprint Registration / Deletion / Search

- **NBioAPI\_AddFIRToNSearchDB**  
Registers a new fingerprint template into the MRFDB.
- **NBioAPI\_SearchDataFromNSearchDB**  
Searches a fingerprint template from the MRFDB, and returns the candidate list after identification.
- **NBioAPI\_IdentifyDataFromNSearchDB**  
Searches a fingerprint template from the MRFDB, and returns the matching result and the fingerprint information if there is a matched template from the MRFDB.
- **NBioAPI\_RemoveDataFromNSearchDB**  
Deletes an existing fingerprint template from the MRFDB.
- **NBioAPI\_RemoveUserFromNSearchDB**



Deletes all fingerprint templates of a user from the MRFDB.

### 3) DB Management for the Memory Resident Fingerprint DB (MRFDB)

- **NBioAPI\_SaveNSearchDBToFile**  
Saves the MRFDB into a file in disk.
- **NBioAPI\_LoadNSearchDBFromFile**  
Loads the file in disk and creates the MRFDB.
- **NBioAPI\_ImportIndexSearchDBToSearchDB**  
Loads the file of IndexSearch DB and creates the MRFDB.
- **NBioAPI\_ClearNSearchDB**  
Deletes all templates from the MRFDB.
- **NBioAPI\_FreeNSearchCandidate**  
Frees the memory allocated for the candidate list after searching. This function must be called after calling NBioAPI\_SearchDataFromNSearchDB.

# 2

## Installation

2.1 System Requirements

2.2 Installation

2.3 Directories & Files

2.4 Demo Program

2.5 License Setup

## 2. Installation

### 2.1 System Requirement

OS	: Windows NT/2000/XP
CPU	: Pentium II 400MHz or above
System Memory	: 256Mb or above
SDK	: NITGEN Fingerprint Device SDK

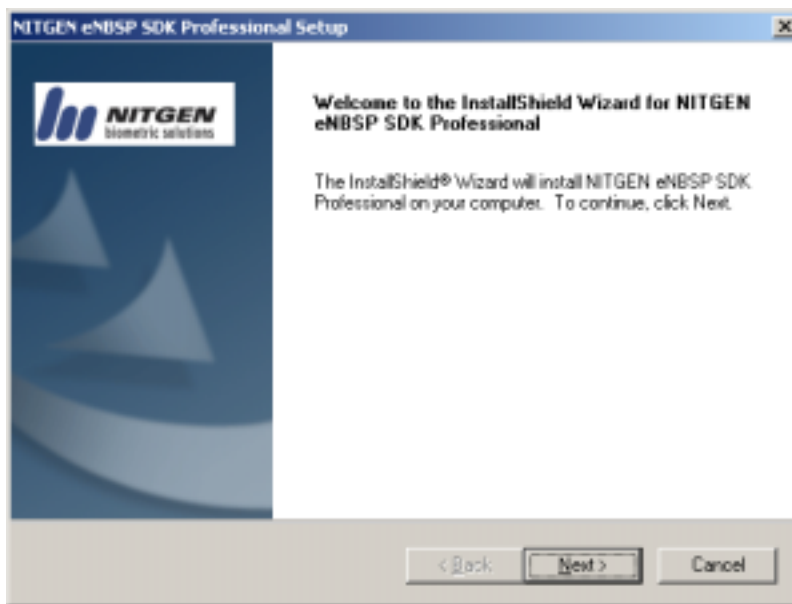
NITGEN SDK must be installed before it can be used, as the NSearch Engine works upon the NITGEN fingerprint devices and the fingerprint extraction algorithm. The NITGEN fingerprint devices consist of PC peripherals, FDP01/02 in parallel type and FDU01 in USB type, and build-in CPU type, FDA01. The NSearch Engine runs properly only if at least one of NITGEN SDK is installed.

The memory requirement is depending on the number of fingerprints that can be installed, because the NSearch Engine loads all fingerprint template data into the memory. The following list describes the minimum RAM requirement depending on the number of fingerprints.

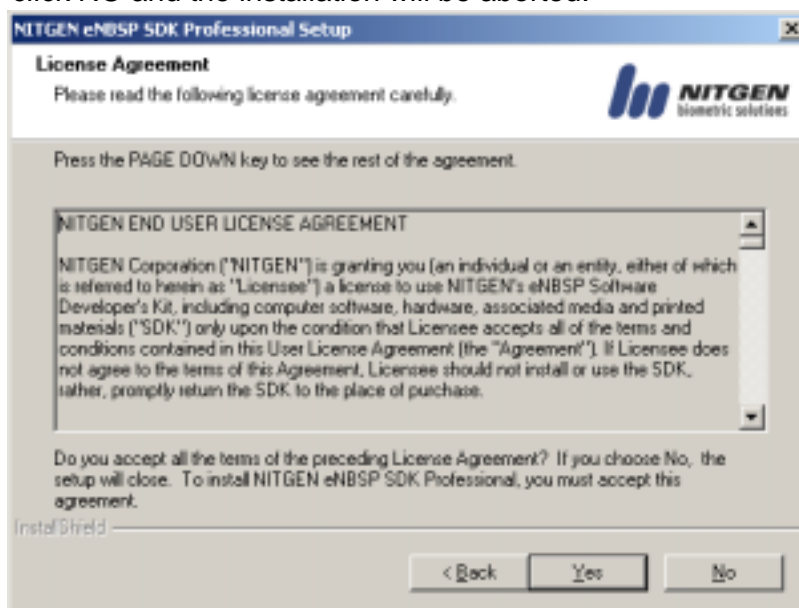
Registered fingerprint No	Required memory size
100	15 Mbyte
1,000	30 Mbyte
2,000	42 Mbyte
5,000	75 Mbyte
10,000	123 Mbyte
20,000	207 Mbyte

## 2.2 Installation

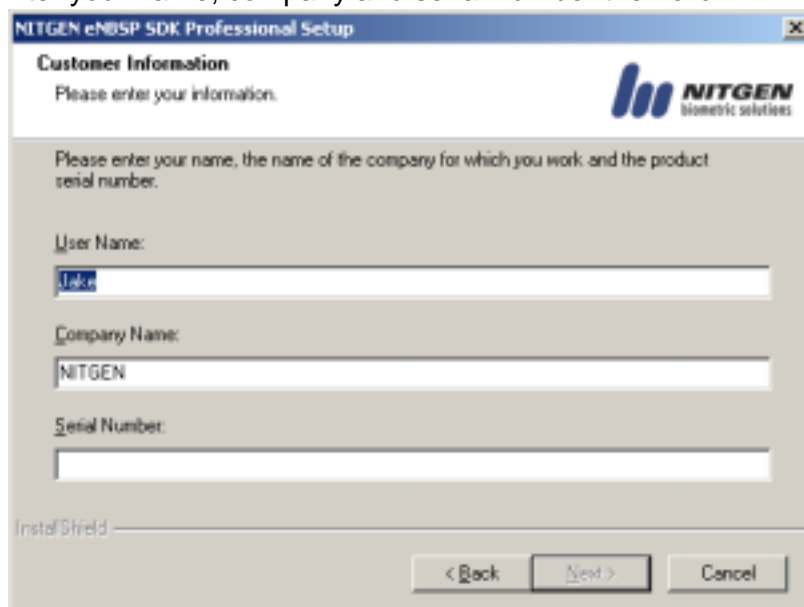
1. Insert installation CD into disk drive.
2. Execute Setup.exe file in the root directory of the CD-ROM drive
3. Read all information displayed in the setup screens, and follow the instructions.
4. Click NEXT to continue.



5. Click YES if you agree to the Software License Agreement. If you do not agree, click NO and the installation will be aborted.

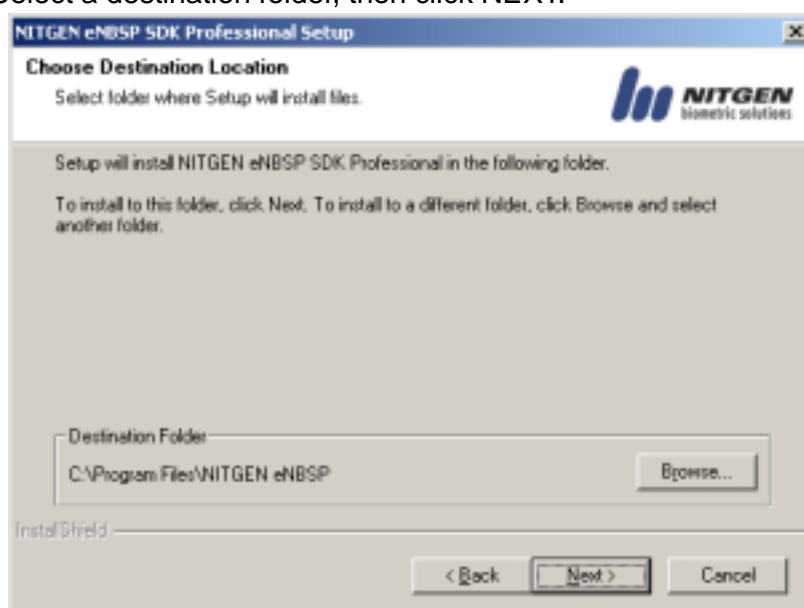


6. Enter your name, company and serial number then click NEXT.



The screenshot shows the 'Customer Information' screen of the 'NITGEN eNBSP SDK Professional Setup' window. The window has a title bar with the text 'NITGEN eNBSP SDK Professional Setup' and a close button. The main area contains the following text: 'Customer Information', 'Please enter your information.', the NITGEN logo, and 'Please enter your name, the name of the company for which you work, and the product serial number.' Below this are three input fields: 'User Name:' with the text 'Jalke', 'Company Name:' with the text 'NITGEN', and 'Serial Number:' which is empty. At the bottom left is a label 'InstallShield' and at the bottom right are three buttons: '< Back', 'Next >', and 'Cancel'.

7. Select a destination folder, then click NEXT.

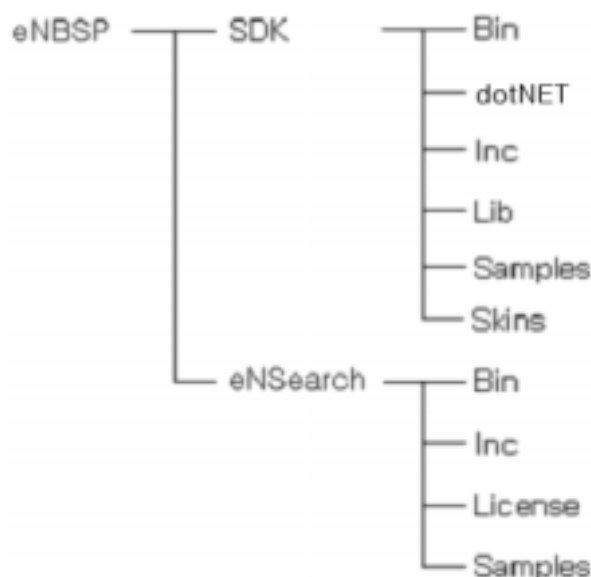


The screenshot shows the 'Choose Destination Location' screen of the 'NITGEN eNBSP SDK Professional Setup' window. The window has a title bar with the text 'NITGEN eNBSP SDK Professional Setup' and a close button. The main area contains the following text: 'Choose Destination Location', 'Select folder where Setup will install files.', the NITGEN logo, and 'Setup will install NITGEN eNBSP SDK Professional in the following folder.' Below this is a paragraph: 'To install to this folder, click Next. To install to a different folder, click Browse and select another folder.' Below the paragraph is a text box labeled 'Destination Folder' containing the text 'C:\Program Files\NITGEN eNBSP'. To the right of the text box is a button labeled 'Browse...'. At the bottom left is a label 'InstallShield' and at the bottom right are three buttons: '< Back', 'Next >', and 'Cancel'.

8. You must register a license file to run NSearch Engine. Please refer to '2.5 License Setup' from detail instructions to issue and install the NSearch Engine license file. If you do not install the license file, all functions on NSearch Engine and sample programs will not work properly. If a license file is not installed, the NSearch engine supports 1,000 fingerprint data for test purpose only.

## 2.3 Directories & Files

After the NSearch SDK installation is complete, the setup files will be copied into the directory structure under C:\Program Files as follows.



The NBioBSP Standard modules are installed into the SDK folder while the Professional modules are into the NSearch folder.

The license file must be issued from NITGEN and copied into 'C:\Program Files\NITGEN eNBSP\eNSearch\License' folder as shown above. Please refer to '2.5 License Setup' for detailed instructions to issue and install the NSearch Engine license file.

### 1) SDK : Library and executable files

This folder contains library files, demo programs and sample source codes for the NBioBSP Standard modules.

### 2) \eNSearch\Bin : Library & executable files

NSearch.dll	- The NSearch Engine main library file
NSearchTest.exe	- Sample program

### 3) \eNSearch\Inc : Header Files

#### 4) \eNSearch\License : License Folder

## 5) \eNSearch\Samples : Sample program

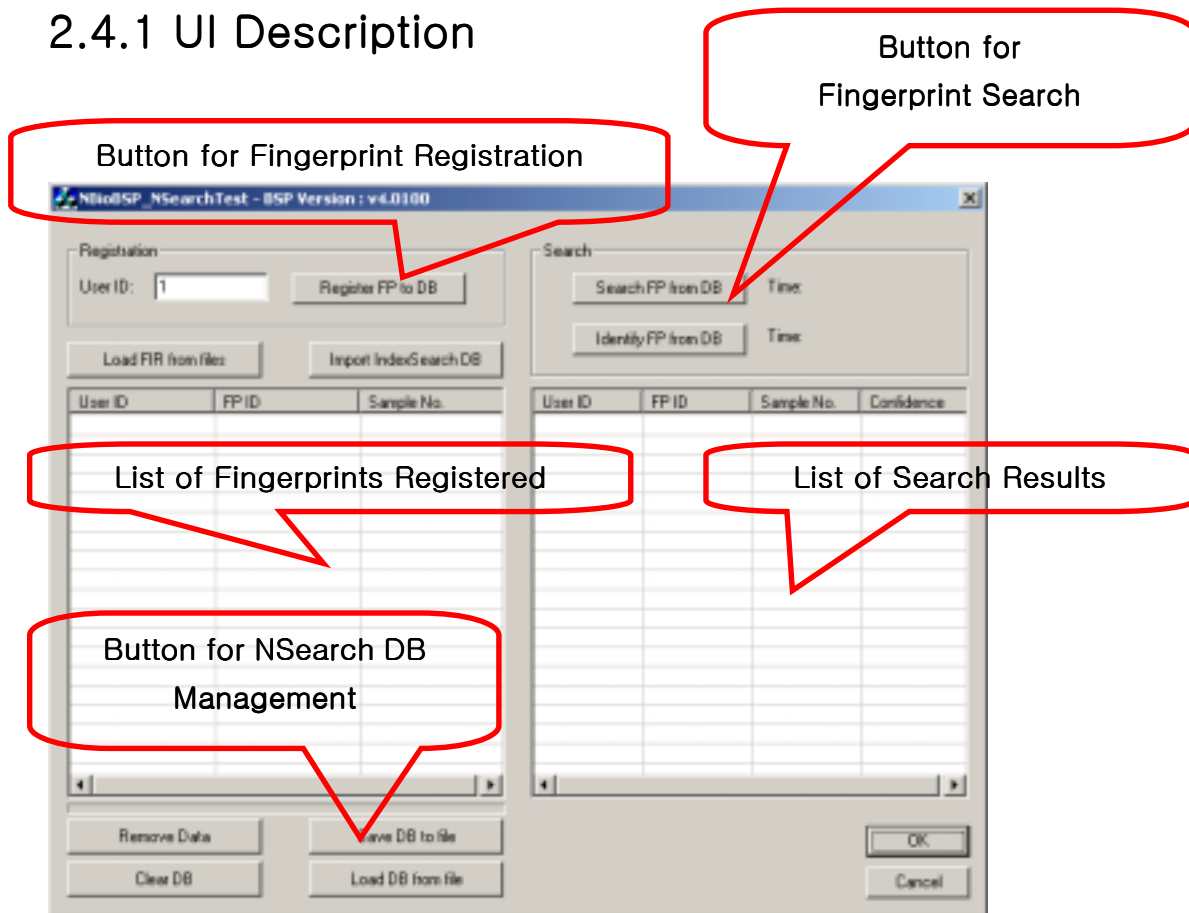
DLL	- Sample folder using the NBioBSP.dll module.
COM	- Sample folder using the COM module, NBioBSPCOM.dll.
dotNET	- Sample folder using the .NET module, NITGEN.SDK.NBioBSP.dll.

VisualC++	- Sample folder containing source codes written in Visual C++ 6.0.
VisualBasic	- Sample folder containing source codes written in Visual Basic 6.0.
Delphi	- Sample folder containing source codes written in Delphi.
C#	- Sample folder containing source codes written in C#.

## 2.4 Demo Program

This program demonstrates the NSearch Engine functions to register fingerprint templates into the Memory Resident Fingerprint DB (MRFDB) and search a template in the MRFDB, using the sample source code.

### 2.4.1 UI Description



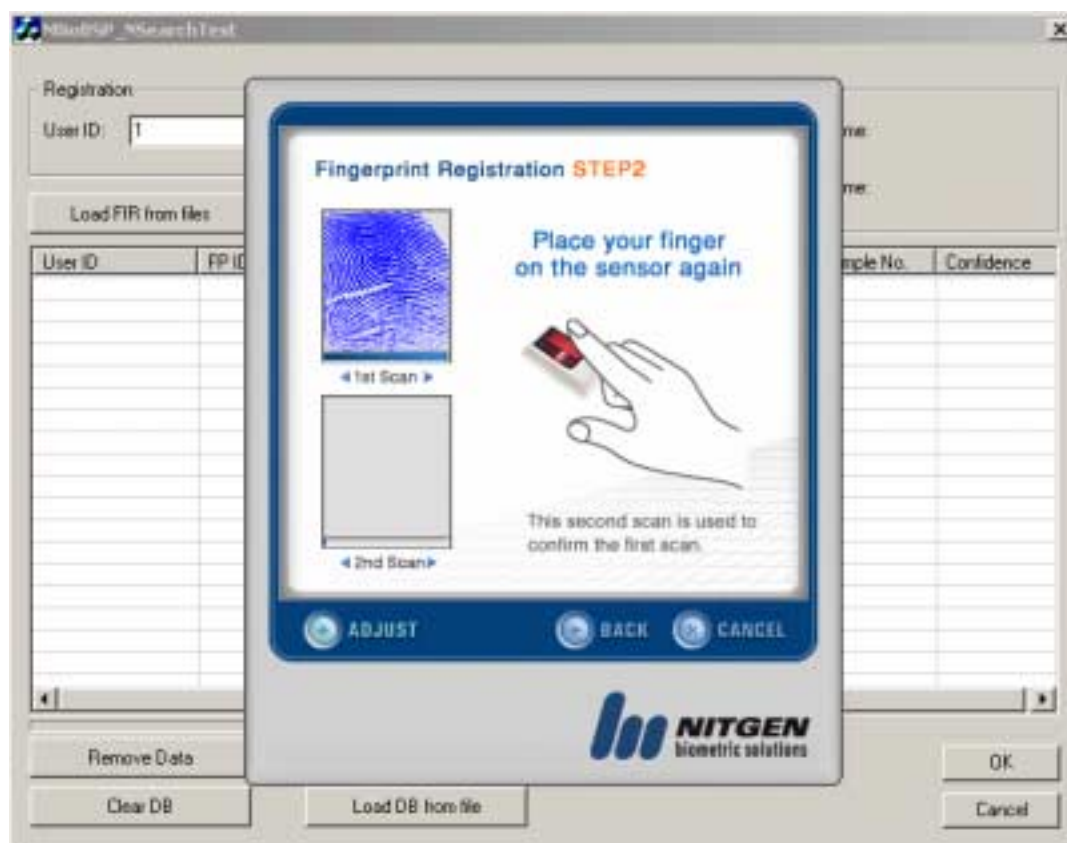
This demo program consists of the function for fingerprint registration and search, the lists of fingerprints registered and search results, and the functions for MRFDB management.

### 2.4.2 Fingerprint Registration

First, enter a user ID in the UserID field and click "Register FP to DB" button, then the



fingerprint registration dialog will prompts. Note that the user ID must be in number,

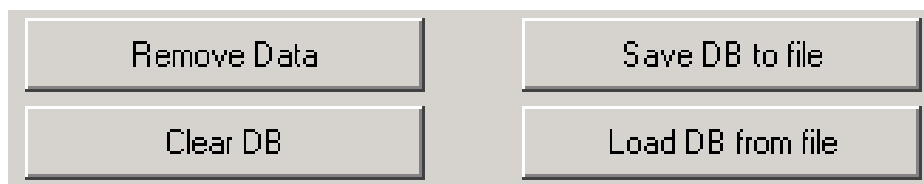


After the fingerprint registration is complete, information about each template is listed up. This list displays two items for one finger registration of a user, since the NBioBSP module asks to input two fingerprint templates to finish registration. For example, the following list indicates that a user, with UserID "1", registered the right thumb (FP ID = 1) twice.

User ID	FP ID	Sample No.
1	1	1
1	1	0

## 2.4.3 DB management

This section demonstrates to initialize the MRFDB and delete a single template from the MRFDB.



### **Remove Data**

Select a list item and click the “Remove Data” button, then the selected item will be deleted from the MRFDB as well as from the list.

### **Clear DB**

Click the “Clear DB” button, then all templates in the MRFDB will be deleted. This function can be used when you want to re-construct the MRFDB.

### **Save DB to file**

The MRFDB may need to be saved into a file and reloaded when necessary. This function can be used before the application is closed, and the MRFDB can be reloaded on the next start of the application.

Clicking “Save DB to file” button asks you select the file name and location, then two files, \*.FDB and \*.FID, are created. The FDB file contains fingerprint data information in binary type and the FID file includes a fingerprint list.

Note that the FID file must be generated by application developers.

### **Load DB from file**

This function is to load the FDB file into the MRFDB. Click the “Load DB from file” button, select a FDB file, then the fingerprint data in the FDB file are loaded into the MRFDB.

## **2.4.4 Search/Identification**

There are two 1:N matching methods, Search and Identification used in the NSearch Engine. The Search function returns 10 candidates with the highest scores, while the Identification determines if the matched fingerprint exists.

### **Search**

This function is to search 10 best matching candidates with the highest scores. The

matching score, called Confidence, ranges from 0 to 9 (highest).

Search

Search FP from DB Time: 0.172 sec

Identify FP from DB Time: Searching Time

Search Button

User ID	FP ID	Sample No.	Confidence
2	2	0	9
2	2	1	9
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Search Results

Any result that the Confidence is zero will not be displayed.

### Identification

This function is to search only one best matching result and returns the user.

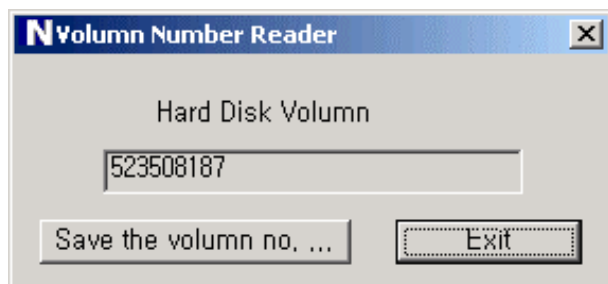
Identify FP from DB Time: 0.47 sec

User ID	FP ID	Sample No.	Confidence
2	3	0	-

The values, Sample No. and Confidence, are not used for Identification.

## 2.5 License Setup

- 1) Execute the Volume Number Reader from Start -> Programs -> NITGEN eNBSP -> eNSearch.



- 2) Click the 'Save the volume no' button to save the hard disk volume number into a text file.
- 3) Send this file, VolumnNo.txt, to NITGEN by email and receive a license file including the expiration date and the maximum number of fingerprints to be used.
- 4) Copy the license file into following location.  
'C:\Program Files\NITGEN eNBSP\eNSearch\License'

**IMPORTANT)** In case of the hard disk drive with 'C:' has been replaced, you must issue the license file again, following the above instructions.



# 3

# SDK Programming

- 3.1 NSearch Engine
- 3.2 System structure
- 3.3 DB structure
- 3.4 NSearch Engine APIs
- 3.5 Visual Basic Programming
- 3.6 Delphi Programming
- 3.7 C#(.NET) Programming

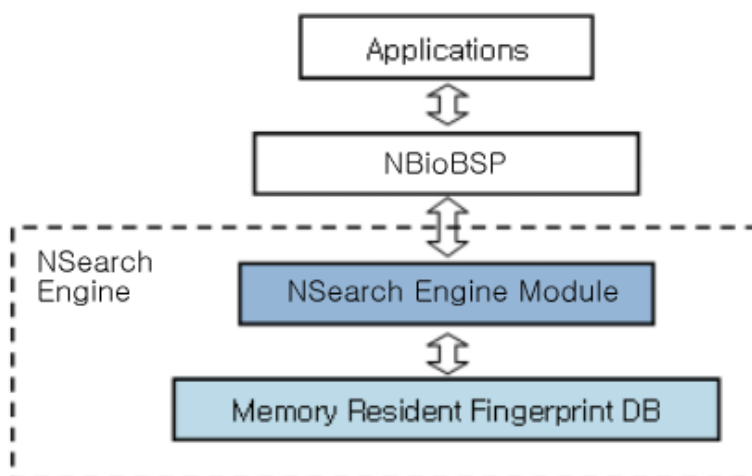
## 3. SDK Programming

This chapter describes how to build the 1:N matching system using NSearch Engine.

### 3.1 NSearch Engine structure

NSearch Engine is controlled by NITGEN NBioBSP module.

The following picture is the module diagram of NSearch Engine.



## 1) Module Description

### **NBioBSP :**

NBioBSP, the Secure Biometric Service Provider from NITGEN, is the interface module between application programs and NSearch Engine. The NBioBSP module is used to capture fingerprints, perform matching operation, and directly control the NSearch Engine.

### **NSearch Engine module:**

NSearch Engine is the main module the NSearch SDK to support large volume of 1:N matching operation. The NSearch Engine registers, manages and searches fingerprint data in Memory Resident Fingerprint DB (MRFDB).

### **Memory Resident Fingerprint DB (MRFDB):**

Memory Resident Fingerprint DB (MRFDB) is the fingerprint database allocated in memory in a hash table form that is reproduced with some additional features for fast matching performance.

## 3.2 System Structure

NSearch Engine can be integrated with the following NITGEN fingerprint device products; NBioBSP SDK, FDx DK, and FDA01 to build an 1:N matching system..

- NBioBSP SDK : NITGEN high-level fingerprint authentication system DK
- FDx DK : NITGEN low-level fingerprint authentication system DK
- FDA01 : NITGEN build-in CPU type fingerprint device

**Note:** In the following diagram, the part of User DB is a system database including user information and additional properties for the application, not related with the Memory Resident DB used in NSearch Engine.

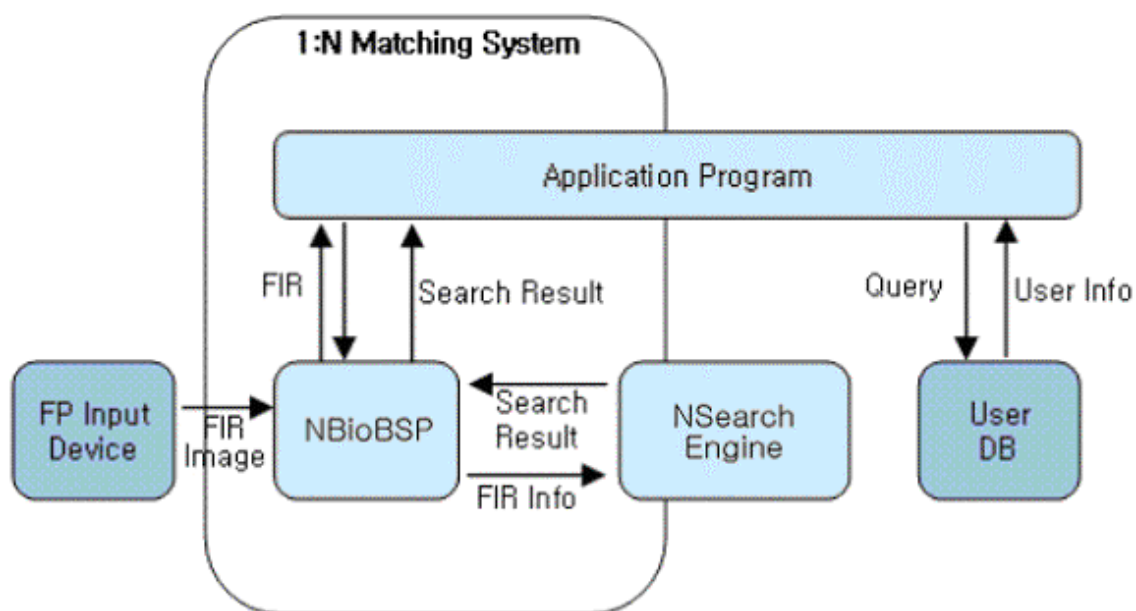
Each diagram consists of two different types of system structure, in stand-alone and client/server type.

The system designed for standalone type contains all devices and modules needed; the fingerprint device, the device driver, the fingerprint extraction module and the NSearch Engine module. All these modules are controlled by an application.

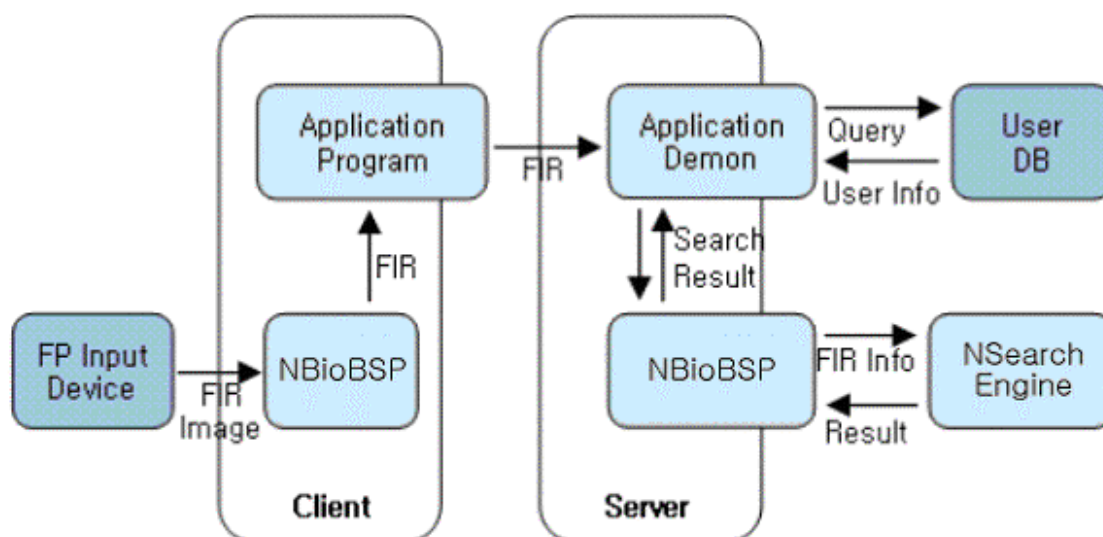
The system for client/server type is composed of two different physical systems. The client system captures and processes fingerprint data while the server performs fingerprint registration and identification.



## 1) System diagram integrated with NBioBSP

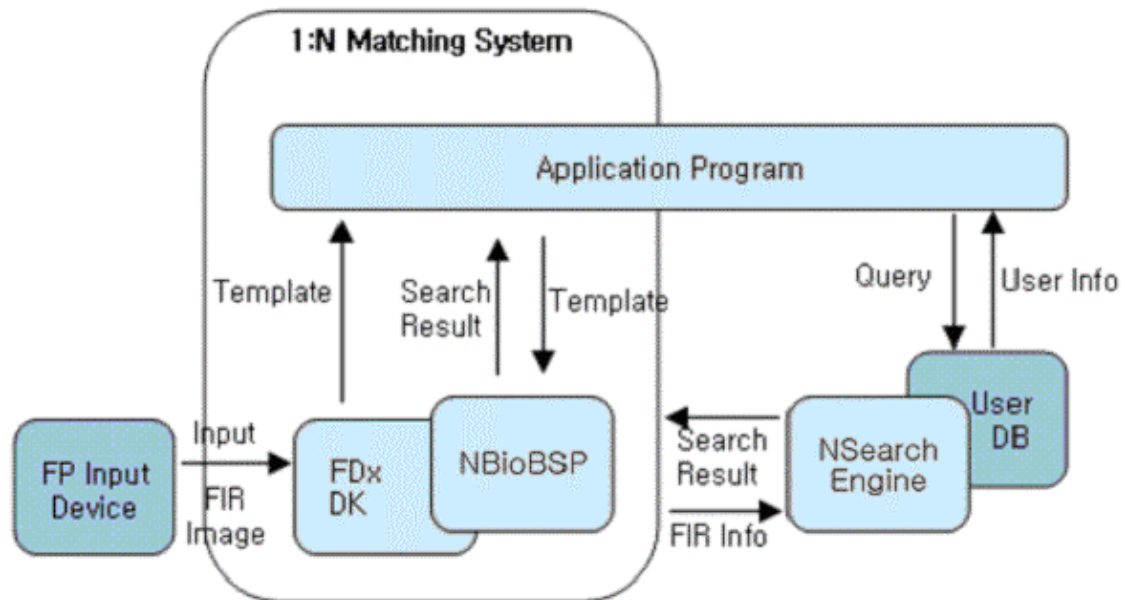


[Standalone type 1:N matching system]

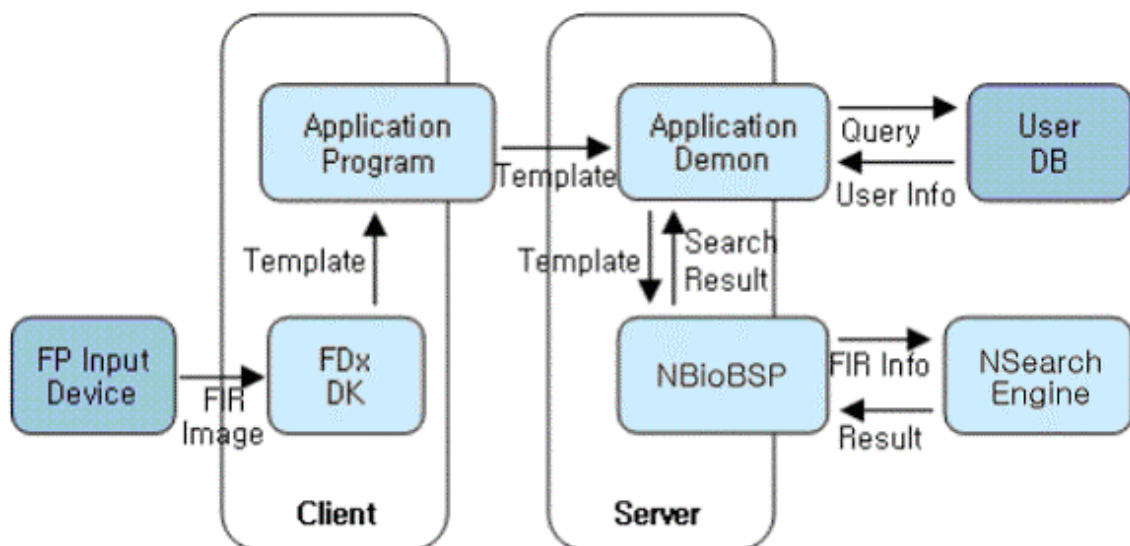


[Client-Server type 1:N matching system]

## 2) System diagram integrated with FDx DK

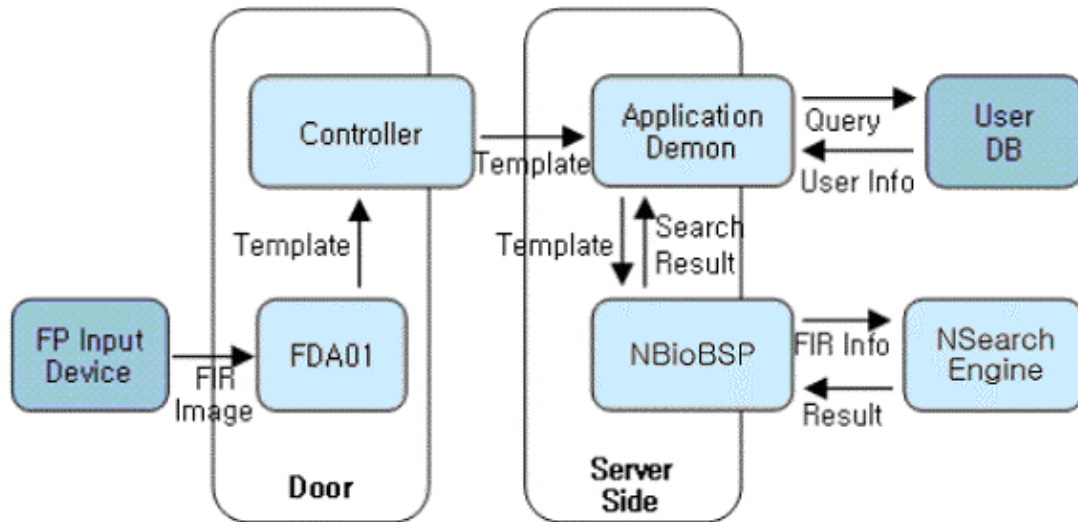


[Standalone type 1:N matching system]



[Client-Server type 1:N matching system]

### 3) System diagram integration with FDA01



[Client-Server type 1:N matching system]

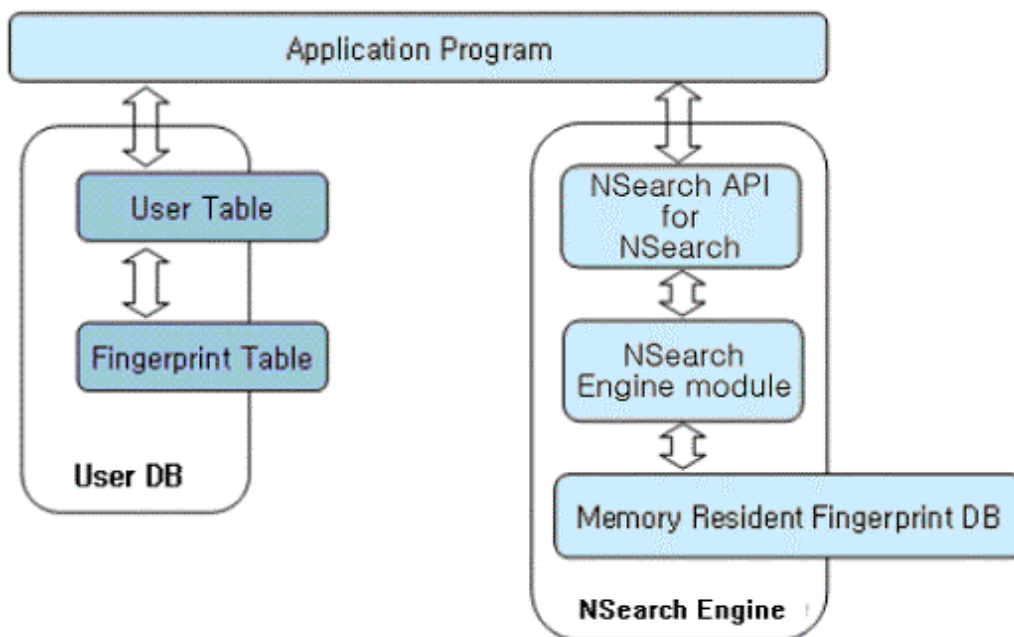
## 3.3 DB structure

### 1) System DB

The followings are database types the 1:N fingerprint matching system may use.

**Memory Resident Fingerprint DB (MRFDB)** : is the fingerprint database in memory in a hash table form reproduced with some additional features for fast matching performance. The MRFDB is controlled by the NSearch Engine internally.

**User DB** : is the system database including user information and additional properties for an application, not related with the Memory Resident DB used in NSearch Engine. The User DB is controlled by application programs.

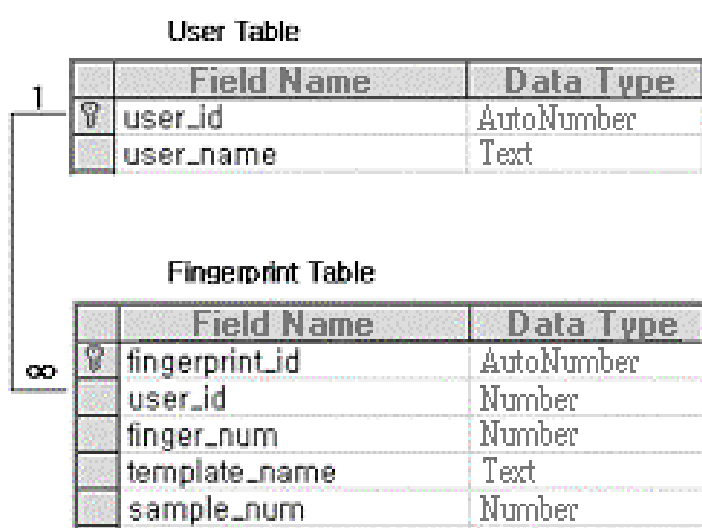


[DB structure of 1:N matching system]

## 2) User DB

User DB means the system database used in applications. The User ID must contain at least following data fields, and developers can add some more fields depending on the purpose of the system. The User DB must have two tables as below, User Table and Fingerprint Table.

### Table structure and relations



### Table fields

**User Table** : stores a record per each user to be registered.

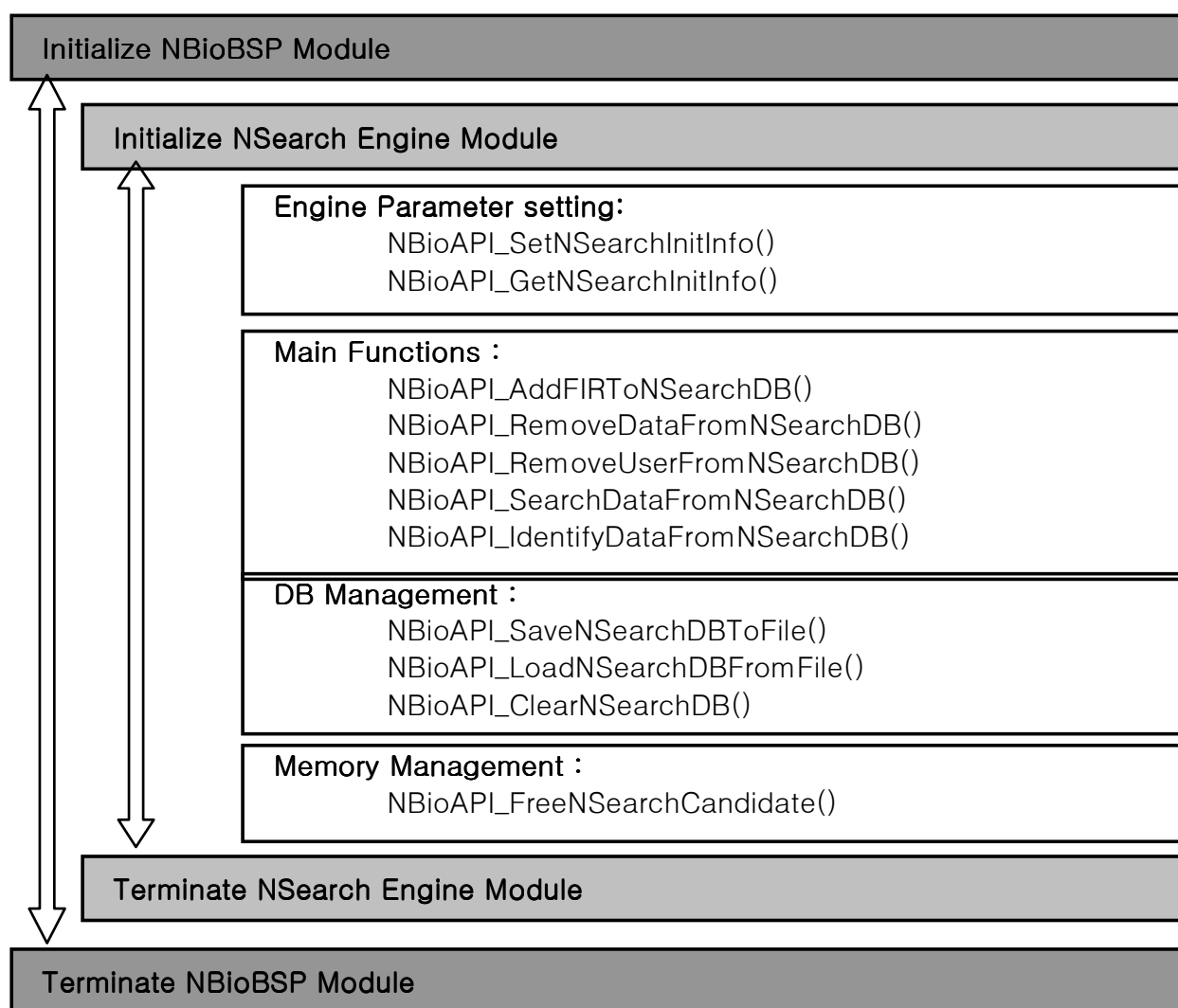
Field Name	Data Type	Description
user_id	Serial Number	As a primary key, it is automatically generated for each user.
user_name	Text	User Name

**Fingerprint Table** : stores a record per each fingerprint data.

Field Name	Data Type	Description
fingerprint_id	Serial Number	As a primary key, it is automatically generated for each fingerprint.
user_id	Number	User ID of the fingerprint (Foreign key in User Table)
finger_num	Number	Fingerprint Number Appendix A.3)
Template_name	Text	Full path of the template file registered
Sample_num	Number	Number of enrolled with same finger

## 3.4 NSearch Engine APIs

In order to control the NSearch Engine, the NBioBSP module provides the following APIs to initialize/terminate the Engine, configure parameter settings, perform main functions, and manage DB and memory. This chapter describes how to use these APIs.



[ NSearch Engine NBioBSP Function Structure ]

### 3.4.1 Initialization/termination/parameter setting

Call NBioAPI\_InitNSearchEngine() function to start the NSearch Engine. This function initializes global variables and allocates Memory Resident Fingerprint DB (MRFDB). Calling NBioAPI\_TerminateNSearchEngine() will free all allocated memory used in the Engine.

#### 1) Initializing NSearch Engine

```
DLLEXPORT DWORD WINAPI NBioAPI_InitNSearchEngine (
    NBioAPI_HANDLE    hHandle);
```

\* DESCRIPTION : Initializes the NSearch Engine.

\* INPUT : NBioBSP Handle

\* OUTPUT : Error code

NBioAPI\_InitNSearchEngine() function is used to initialize the NSearch Engine. While initialization, this function internally checks license validation if expired or exceeding the maximum number of fingerprints. If the license is invalid, the initialization will fail. The Engine parameters are as below.

```
typedef struct nbioapi_NSearch_init_info_0 {
    NBioAPI_UINT32  StructureType;           /* must be 0 */
    NBioAPI_UINT32  MaxCandidateNumber;     /* Default = 10 */
    NBioAPI_UINT32  researved0;             /* Reserved */
    NBioAPI_UINT32  researved1;             /* Reserved */
    NBioAPI_UINT32  researved2;             /* Reserved */
    NBioAPI_UINT32  researved3;             /* Reserved */
    NBioAPI_UINT32  researved4;             /* Reserved */
    NBioAPI_UINT32  researved5;             /* Reserved */
    NBioAPI_UINT32* researved6;             /* Reserved */
} NBioAPI_NSEARCH_INIT_INFO_0, *NBioAPI_NSEARCH_INIT_INFO_PTR_0;
```

The MaxCandidateNumber member in this structure indicates the number of candidates to be searched for best matching fingerprint data using by NSearch Engine. For example, specifying 10 in the MaxCandidateNumber member of NBioAPI\_NSEARCH\_INIT\_INFO\_0 structure means that 10 candidates will be returned.



To update this value, call NBioAPI\_SetNSearchInitInfo() function.

```
If (NBioAPI_InitNSearchEngine (m_hNBioBSP) != NBioAPIERROR_NONE)
{
    // Init module failed. Show error message.
}
// Init success.
```

## 2) Closing NSearch Engine

```
DLLEXPORT DWORD WINAPI NBioAPI_TerminateNSearchEngine(
    NBioAPI_HANDLE    hHandle);
```

\* DESCRIPTION : Closes the NSearch Engine and frees all allocation.

\* OUTPUT : Error code

Call NBioAPI\_TerminateNSearchEngine() function to close the NSearch Engine. This function must be called to free all allocated memory used in the Engine.

Note that this is done automatically when NBioAPI\_Terminate() function is called.

```
NBioAPI_RETURN ret;
ret = NBioAPI_TerminateNSearchEngine ();    // Terminate NSearch Module
```

## 3) Reading NSearch Engine parameters

```
DLLEXPORT DWORD WINAPI NBioAPI_GetNSearchInitInfo(
    NBioAPI_HANDLE    hHandle,
    NBioAPI_UINT8     nStructureType,
    NBioAPI_INIT_INFO_PTR    pInitInfo);
```

\* DESCRIPTION : Reads the NSearch Engine parameters.

\* INPUT : Structure type and value to read.

\* OUTPUT : Error code, NBioAPI\_INIT\_INFO structure.

NBioAPI\_GetNSearchInitInfo() function is used to read the NSearch Engine parameters. These parameters can only be read after the Engine initialization.

```
NBioAPI_NSEARCH_INIT_INFO_0    initInfo0; // NSearch Engine Parameter.  
NBioAPI_RETURN                 ret;
```

```
ret = NBioAPI_GetNSearchInitInfo(m_hNBioBSP, 0, &initInfo0); // Get Engine  
parameter
```

#### 4) Setting NSearch Engine parameters

```
DLLEXPORT DWORD WINAPI NBioAPI_SetNSearchInitInfo(  
    NBioAPI_HANDLE          hHandle,  
    NBioAPI_UINT8           nStructureType,  
    NBioAPI_INIT_INFO_PTR   pInitInfo);
```

- \* DESCRIPTION : Sets the NSearch Engine parameters.
- \* INPUT : Structure type to set.
- \* OUTPUT : Error code.

NBioAPI\_SetNSearchInitInfo() function is to set the NSearch Engine parameters. These parameters can only be updated after the Engine initialization.

```
NBioAPI_NSEARCH_INIT_INFO_0    initInfo0; // NSearch Engine Parameter.  
NBioAPI_RETURN                 ret;
```

```
memset(&initInfo0, 0, sizeof(NBioAPI_NSEARCH_INIT_INFO_0));  
initInfo0.StructureType = 0;  
initInfo0.MaxCandidateNumber = 20;  
ret = NBioAPI_SetNSearchInitInfo(m_hNBioBSP, 0, &initInfo0); // Set Engine  
parameter
```

## 3.4.2 Fingerprint Registration / Deletion / Search

The main functions of the NSearch Engine are to register, delete and search a fingerprint data. While fingerprint registration or identification, any of the NBioBSP FIR format can be used.

### 1) Registering fingerprint into the MRFDB

```
DLLEXPORT DWORD WINAPI NBioAPI_AddFIRToNSearchDB(  
    NBioAPI_HANDLE          hHandle,  
    const NBioAPI_INPUT_FIR_PTR pInputFIR,  
    NBioAPI_UINT32          nUserID,  
    NBioAPI_NSEARCH_SAMPLE_INFO_PTR pSampleInfo);
```

\* DESCRIPTION : Registers a fingerprint data into the MRFDB.

\* INPUT : NBioAPI\_INPUT\_FIR(FIR Handle, Full FIR, Text FIR) & User ID.

\* OUTPUT : Error code, pSampleInfo

Call NBioAPI\_AddFIRToNSearchDB() function to register fingerprint templates into the MRFDB. The FIR can contain more than one fingerprint data for a user, thus single user registration could generate several template data into the MRFDB. Refer to pSampleInfo for more information about each template data.

```
typedef struct nbioapi_nsearch_sample_info  
{  
    NBioAPI_UINT32    ID;  
    NBioAPI_UINT8     SampleCount[11];  
} NBioAPI_NSEARCH_SAMPLE_INFO;
```

The ID member of NBioAPI\_NSEARCH\_SAMPLE\_INFO structure represents the user ID and the SampleCount member contains the number of templates for each finger. The following example shows that the SampleCount array contains 2 right thumb and 2 left thumb fingers registered.

```
SampleCount[0] = 0;      //NBioAPI_FINGER_ID_UNKNOWN
SampleCount[1] = 2;    //NBioAPI_FINGER_ID_RIGHT_THUMB
SampleCount[2] = 0;      //NBioAPI_FINGER_ID_RIGHT_INDEX
SampleCount[3] = 0;      //NBioAPI_FINGER_ID_RIGHT_MIDDLE
SampleCount[4] = 0;      //NBioAPI_FINGER_ID_RIGHT_RING
SampleCount[5] = 0;      //NBioAPI_FINGER_ID_RIGHT_LITTLE
SampleCount[6] = 2;    //NBioAPI_FINGER_ID_LEFT_THUMB
SampleCount[7] = 0;      //NBioAPI_FINGER_ID_LEFT_INDEX
SampleCount[8] = 0;      //NBioAPI_FINGER_ID_LEFT_MIDDLE
SampleCount[9] = 0;      //NBioAPI_FINGER_ID_LEFT_RING
SampleCount[10] = 0;     //NBioAPI_FINGER_ID_LEFT_LITTLE
```

## 2) Deleting fingerprint

### A. Deleting one template from the MRFDB

```
DLLEXPORT DWORD WINAPI NBioAPI_RemoveDataFromNSearchDB(
    NBioAPI_HANDLE          hHandle,
    NBioAPI_NSEARCH_FP_INFO_PTR pFpInfo);
```

- \* DESCRIPTION : Deletes a template data from the MRFDB.
- \* INPUT : Template information to delete.
- \* OUTPUT : Error code

NBioAPI\_RemoveDataFromNSearchDB() function is to delete a template from the MRFDB. This function deletes only one template. In order to delete all templates for a user, call NBioAPI\_RemoveUserFromNSearchDB() function.

Note that you must also delete the corresponding data in the system database in your application to maintain data synchronization with the MRFDB.

```
NBioAPI_RETURN ret;          // error code
NBioAPI_NSEARCH_FP_INFO infoFP; // information of fingerprint to be removed
```

```
InfoFP.ID = 1;  
InfoFP.FingerNumber = NBioAPI_FINGER_ID_RIGHT_THUMB;  
InfoFP.SampleNumber = 0;  
ret = NBioAPI_RemoveDataFromNSearchDB(m_hNBioBSP, &infoFP);
```

#### B. Deleting all templates for a user from the MRFDB

```
DLLEXPORT DWORD WINAPI NBioAPI_RemoveUserFromNSearchDB(  
    NBioAPI_HANDLE          hHandle,  
    NBioAPI_UINT32          nUserID);
```

- \* DESCRIPTION : Deletes all templates for a user from the MRFDB.
- \* INPUT : User ID to delete
- \* OUTPUT : Error code

Call NBioAPI\_RemoveUserFromNSearchDB() function to delete all templates for a user from the MRFDB. Use NBioAPI\_RemoveDataFromNSearchDB() function to delete a single template.

Note that you must also delete the corresponding data in the system database in your application to maintain data synchronization with the MRFDB.

```
NBioAPI_RETURN ret;          // error code  
NBioAPI_UINT32 nUserID;      // ID of user to be removed
```

```
nUserID = 1;  
ret = NBioAPI_RemoveUserFromNSearchDB(m_hNBioBSP, nUserID);
```

### 3) Search/Identification

#### A. Search (Candidate listing)

```
DLLEXPORT DWORD WINAPI NBioAPI_SearchDataFromNSearchDB (  
    NBioAPI_HANDLE          hHandle,
```

```
const NBioAPI_INPUT_FIR_PTR      pInputFIR,
NBioAPI_NSEARCH_CANDIDATE_PTR*  ppCandidate,
NBioAPI_VOID_PTR                pReserved = NULL);
```

\*DESCRIPTION : Search the memory DB and create a candidate list.  
 \*INPUT : FIR data to search, Pointer of pointer for Candidate List.  
 \*OUTPUT : Candidate list ,Error code

NBioAPI\_SearchDataFromNSearchDB() function is used to search the template candidates that are most closely matched with the input template in the MRFDB. In order to search with the FIR Handle, input NBioAPI\_FIR\_HANDLE in the Form of NBioAPI\_INPUT\_FIR, and input the FIR Handle in the inputFIRinBSP member, then call the NBioAPI\_SearchDataFromNSearchDB() function. Then, the candidate list will be returned including the FingerNumber, SampleNumber, and ConfidenceLevel.

```
typedef struct nbioapi_NSearch_candidate {
    NBioAPI_UINT32    ID;
    NBioAPI_UINT8     FingerID;
    NBioAPI_UINT8     SampleNumber;
    NBioAPI_UINT8     ConfidenceLevel;
} NBioAPI_NSEARCH_CANDIDATE;
```

Calling NBioAPI\_FreeNSearchCandidate() function will free the memory allocated for the candidate list.

```
NBioAPI_RETURN      ret;          // error code
NBioAPI_INPUT_FIR   inputFIR;    // input fingerprint data
NBioAPI_NSEARCH_CANDIDATE_PTR pCandidate; // output candidate list
```

```
inputFIR.Form = NBioAPI_FIR_FORM_FULLFIR;
inputFIR.InputFIR.FIR = &m_myFIR;
ret = NBioAPI_SearchDataFromNSearchDB
    (m_hNBioBSP, &inputFIR, &pCandidate, NULL); // Search fingerprint from DB
```

## B. Identification

```
DLLEXPORT DWORD WINAPI NBioAPI_IdentifyDataFromNSearchDB(
    NBioAPI_HANDLE                hHandle,
    const NBioAPI_INPUT_FIR_PTR    pInputFIR,
    NBioAPI_FIR_SECURITY_LEVEL     nSecuLevel,
    NBioAPI_NSEARCH_FP_INFO_PTR    pFpInfo);
```

\*DESCRIPTION : Identifies a fingerprint data in memory DB.

\*INPUT : FIR data to identify, Security level.

\*OUTPUT : Template information identified, Error code

NBioAPI\_IdentifyDataFromNSearchDB() function is to identify a template from the MRFDB. Input the FIR for the second parameter and the security level for the third parameter, then the identification result will be returned. If the identification is successful, the pFpInfo parameter in the NBioAPI\_NSEARCH\_FP\_INFO structure is filled up with fingerprint information matched, otherwise, it remains empty. The fingerprint information contains ID, FingerID and SampleNumber.

```
typedef struct nbioapi_NSearch_fp_info {
    NBioAPI_UINT32    ID;
    NBioAPI_UINT8     FingerID;
    NBioAPI_UINT8     SampleNumber;
} NBioAPI_NSEARCH_FP_INFO;
```

The security level, in range from 0 to 9, determines the threshold that can be used to match a template against the other. Only if the ConfidenceLevel is higher than the security level setting, the matching result will be successful.

```
NBioAPI_RETURN    ret;        // error code
NBioAPI_INPUT_FIR    inputFIR; // input fingerprint data
NBioAPI_NSEARCH_FP_INFO infoFP; // output fingerprint information
```

```
inputFIR.Form = NBioAPI_FIR_FORM_FULLFIR;
```

```
inputFIR.InputFIR.FIR = &m_myFIR;  
ret = NBioAPI_IdentifyDataFromNSearchDB  
      (m_hNBioBSP, &inputFIR, 5, &infoFP); // Identify fingerprint from DB
```



### 3.4.3 MRFDB Management

When there is any change in the MRFDB, the system database must be updated with the corresponding data to maintain data synchronization with the MRFDB. In addition, since the MRFDB, allocated in the memory, will be lost when the application is closed, the MRFDB must be saved into a file if it needs to be reused when the application restarts,.

#### 1) Saving MRFDB into a file

```
DLLEXPORT DWORD WINAPI NBioAPI_SaveNSearchDBToFile(  
    NBioAPI_HANDLE          hHandle,  
    const NBioAPI_CHAR*     szFilepath)
```

\* DESCRIPTION : Saves the MRFDB to disk.  
\* INPUT : Full path name of DB file name  
\* OUTPUT : Error code

Call NBioAPI\_SaveNSearchDBToFile() function to save the MRFDB into a file in disk.  
Input the full path name for the thirist parameter.

```
NBIOAPI_RETURN ret; // error code  
NBioAPI_CHAR szDBName[MAX_PATH]; // full-path file name
```

```
strcpy(szDBName, "c:\\Data\\MyDB.dat");  
ret = NBioAPI_SaveNSearchDBToFile(m_hNBioBSP, szDBName);
```

#### 2) Loading MRFDB from a file

```
DLLEXPORT DWORD WINAPI NBioAPI_LoadNSearchDBFromFile(  
    NBioAPI_HANDLE          hHandle,  
    const NBioAPI_CHAR*     szFilepath)
```

\* DESCRIPTION : Loads the MRFDB from disk.

\* INPUT : Full path name of DB file name  
\* OUTPUT : Error code

Call NBioAPI\_LoadNSearchDBFromFile() function to load a file from disk into the MRFDB. This function can be called after NSearch Engine restarts.

```
NBIOAPI_RETURN ret; // error code  
NBioAPI_CHAR szDBName[MAX_PATH]; // full-path file name
```

```
strcpy(szDBName, "c:\\Data\\MyDB.dat");  
ret = NBioAPI_LoadNSearchDBToFile(m_hNBioBSP, szDBName);
```

### 3) Clearing the MRFDB

```
DLLEXPORT DWORD WINAPI NBioAPI_ClearNSearchDB(  
    NBioAPI_HANDLE hHandle);
```

\* DESCRIPTION : Clears all data in the MRFDB.  
\* INPUT : None  
\* OUTPUT : Error code

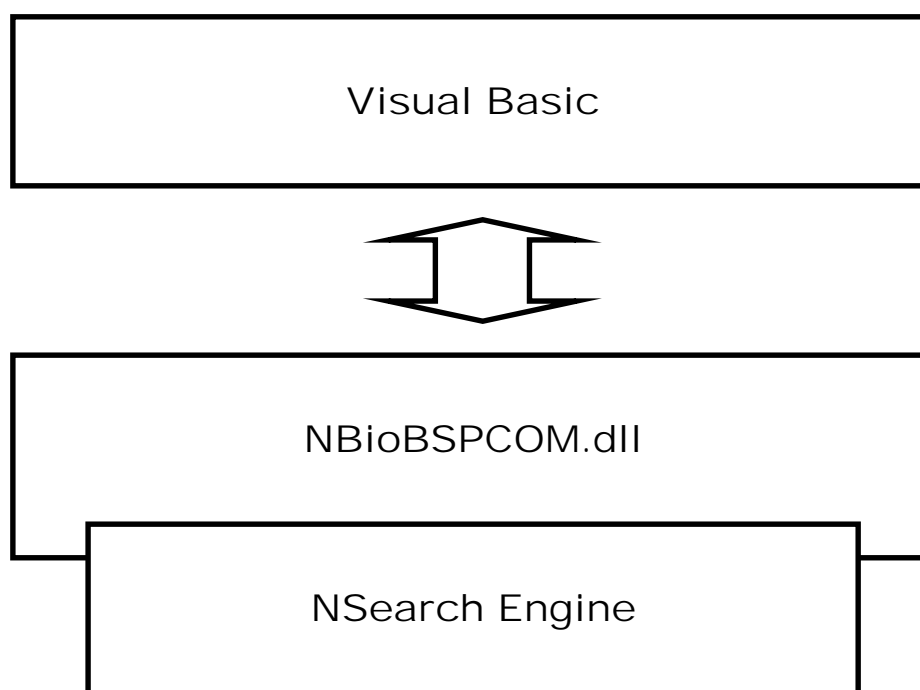
To delete all template data in the MRFDB, Call NBioAPI\_ClearNSearchDB() function.

```
NBIOAPI_RETURN ret; // error code
```

```
ret = NBioAPI_ClearNSearchDB(m_hNBioBSP);
```

## 3.5 Visual Basic Programming

The NBIOBSPCOM.dll can be used when programming in the Visual Basic development environment. This module, in COM interfaces, is designed for web developers and for those using RAD tools such as Visual Basic or Delphi



[ NBioBSPCOM.dll and NSearch Engine Structure ]

### 3.5.1 Initialization / termination / parameter setting

Declare a new object for NBIOBSPCOM module to start the NSearch Engine. This method initializes global variables and allocates the Memory Resident Fingerprint DB (MRFDB). Declaring the object free will free all allocated memory used in the Engine.

#### 1) Initializing the NSearch Engine object

Declare a new object to initialize the NSearch Engine.

```
Dim objNBioBSP as NBIOBSPCOMLib.NBioBSP
...
Set objNBioBSP = New NBIOBSPCOMLib.NBioBSP
Set objNSearch = objNBioBSP.NSearch
...
```

When the object is created, the NSearch Engine is automatically initialized. While initialization, this method internally checks license validation if expired or exceeding the maximum number of fingerprints. If the license is invalid, the initialization will fail.

Reading and updating parameters used in the NSearch Engine are only possible after initialization.

```
objNSearch.MaxCandidateNumber = 5
If objNSearch.ErrorCode <> NBioAPIERROR_NONE then
    // Init module failed. Show error message.
    // objNSearch.ErrorDescript
Else
    // Init success.
End if
```

#### 2) Closing the NSearch Engine object

Declare the object free to terminate the NSearch Engine. This needs to be done before the application is closed.

```
Set objNSearch = nothing
```

```
Set objNBioBSP = nothing
```

### **3) Reading the NSearch Engine parameters**

The MaxCandidateNumber property can be used to read the number of candidates to be generated from the NSearch Engine.

```
ret = objNSearch.MaxCandidateNumber
```

### **4) Setting the NSearch Engine parameters**

The MaxCandidateNumber property can be specified to update the number of candidates to be generated from the NSearch Engine.

```
objNSearch.MaxCandidateNumber = 5
```

## 3.5.2 Fingerprint Registration / Deletion / Search

The main functions of the NSearch Engine are to register, delete and search a fingerprint data. While fingerprint registration or identification, the NBioBSP FIR can be used only in text format.

### 1) Registering fingerprint into the MRFDB

AddFIR(BSTR bszInputFIR, LONG nUserID)

This method is to register fingerprint templates into the MRFDB. The FIR can contain more than one fingerprint data for a user, thus single user registration could generate several template data into the MRFDB. Refer to the properties for more information about each template data.

```
Dim objResult As ICandidateList
Dim nFingerID As Long
' Get FIR data
Set objDevice = objNBioBSP.Device
Set objExtraction = objNBioBSP.Extraction

Call objDevice.Open(NBioBSP_DEVICE_ID_AUTO_DETECT)
Call objExtraction.Enroll(Null)
Call objDevice.Close(NBioBSP_DEVICE_ID_AUTO_DETECT)

szFir = objExtraction.TextEncodeFIR

' Register FIR to NSearch DB
Call objNSearch.AddFIR (szFIR, nUserID)

For Each objResult In objNSearch
    // Print FP information in FIR
    nFingerID = objResult.FingerID
Next
```

In this example code, the objResult, ICandidateList object, contains registration results for each fingerprint. The objResult also includes UserID, FingerID and SampleNumber. The FingerID represents 1 through 10 as right thumb to left little finger, and the SampleNumber can be 0 or 1 as first or second. If using the Capture method, unlike the Enroll method, the SampleNumber will be always 0 value.

Each FingerID values are as follows.

- 0 : NBioAPI\_FINGER\_ID\_UNKNOWN
- 1 : NBioAPI\_FINGER\_ID\_RIGHT\_THUMB
- 2 : NBioAPI\_FINGER\_ID\_RIGHT\_INDEX
- 3 : NBioAPI\_FINGER\_ID\_RIGHT\_MIDDLE
- 4 : NBioAPI\_FINGER\_ID\_RIGHT\_RING
- 5 : NBioAPI\_FINGER\_ID\_RIGHT\_LITTLE
- 6 : NBioAPI\_FINGER\_ID\_LEFT\_THUMB
- 7 : NBioAPI\_FINGER\_ID\_LEFT\_INDEX
- 8 : NBioAPI\_FINGER\_ID\_LEFT\_MIDDLE
- 9 : NBioAPI\_FINGER\_ID\_LEFT\_RING
- 10 : NBioAPI\_FINGER\_ID\_LEFT\_LITTLE

## **2) Deleting fingerprint**

### **A. Deleting one template from the MRFDB**

RemoveData(LONG nUserID, LONG nFingerID, LONG nSampleNumber)

RemoveData method is to delete a template from the MRFDB. This method deletes only one template. In order to delete all templates for a user, use RemoveUser method..

Note that you must also delete the corresponding data in the system database in your application to maintain data synchronization with the MRFDB.

nUserID = 2

nFingerID = 1 ' NBioAPI\_FINGER\_ID\_RIGHT\_THUMB

```
nSampleNumber = 0
```

```
Call objNSearch.RemoveData(nUserID, nFingerID, nSampleNumber)
```

```
If objNSearch.ErrorCode = 0 then
```

```
    ' Success
```

```
Else
```

```
    ' Fail
```

```
End if
```

## **B. Deleting all templates for a user from the MRFDB**

```
RemoveUser (LONG nUserID)
```

Call RemoveUser method to delete all templates for a user from the MRFDB. Use RemoveData method to delete a single template.

Note that you must also delete the corresponding data in the system database in your application to maintain data synchronization with the MRFDB.

```
nUserID = 1
```

```
Call objNSearch.RemoveUser (nUserID)
```

```
If objNSearch.ErrorCode = 0 then
```

```
    ' Success
```

```
Else
```

```
    ' Fail
```

```
End if
```

## **3) Search/Identification**

### **A. Search (Candidate listing)**

```
SearchData (BSTR bszInputFIR)
```

```
...
```



Use the SearchData method to search the template candidates that are most closely matched with the input template in the MRFDB. Input the FIR data, then the candidate list will be returned including the UserID, FingerID, SampleNumber and ConfidenceLevel.

```
Dim objResult As ICandidateList ' CandidateList or Result object
' Capture fingerprint
...
Call objNSearch.SearchData(szFir)

If objNSearch.ErrorCode = 0 then
    For Each objResult In objNSearch
        Set ListItem = ListResult.ListItems.Add
        ListItem.Text = objResult.UserID
        ListItem.SubItems(1) = objResult.FingerID
        ListItem.SubItems(2) = objResult.SampleNumber
        ListItem.SubItems(3) = objResult.ConfidenceLevel
        Set ListItem = Nothing
    Next
End if
```

## B. Identification

IdentifyUser (BSTR bszInputFIR, LONG nSecuLevel)

Use the IdentifyUser method to identify a template from the MRFDB. Input the FIR data and the security level, then the identification result will be returned. If the identification is successful, the UserID property is returned with a value, otherwise, it remains empty.

The security level, in range from 0 to 9, determines the threshold that can be used to match a template against the other. Only if the ConfidenceLevel is higher than the security level setting, the matching result will be successful.

```
' Capture fingerprint
...
```

Call objNSearch.IdentifyUser (szFir, nSecuLevel)

If objNSearch.ErrorCode = 0 then

    nUserID               = objNSearch.UserID

End if

### 3.5.3 MRFDB Management

When there is any change in the MRFDB, the system database must be updated with the corresponding data to maintain data synchronization with the MRFDB. In addition, since the MRFDB, allocated in the memory, will be lost when the application is closed, the MRFDB must be saved into a file if it needs to be reused when the application restarts,.

#### 1) Saving the MRFDB into a file

```
SaveDBToFile(BSTR bszFilePath)
```

Use the SaveDBToFile method to save the MRFDB into a file in disk. Input the full path name for the parameter.

```
szFileName = "C:\\data\\MyDB.fdb"  
Call objNSearch.SaveDBToFile(szFileName)
```

```
If objNSearch.ErrorCode = 0 then  
    ' Success  
Else  
    ' Fail  
End if
```

#### 2) Loading the MRFDB from a file

```
LoadDBFromFile(BSTR bszFilePath)
```

Use the LoadDBFromFile method to load a file from disk into the MRFDB. This method can be called after the NSearch Engine restarts.

```
szFileName = "C:\\data\\MyDB.fdb"  
Call objNSearch.LoadDBFromFile(szFileName)
```

```
If objNSearch.ErrorCode = 0 then
```

```
        ' Success  
    Else  
        ' Fail  
    End if
```

### **3) Clearing the MRFDB**

```
ClearDB()
```

To delete all template data in the MRFDB, Use the ClearDB method.

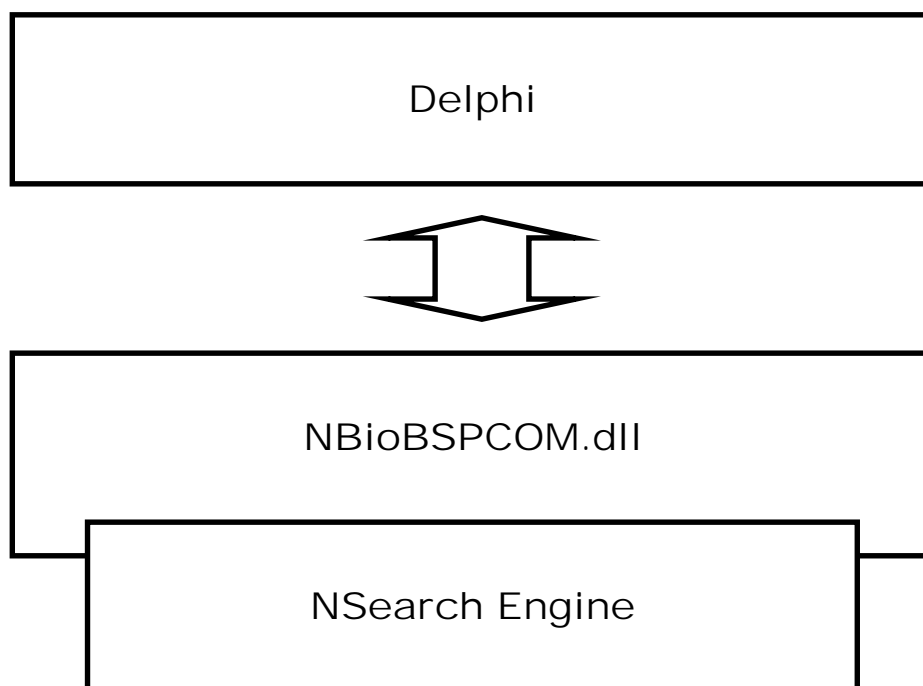
```
Call objNSearch.ClearDB()
```

```
If objNSearch.ErrorCode = 0 then  
    ' Success  
Else  
    ' Fail  
End if
```



## 3.6 Delphi Programming

The NBIOBSPCOM.dll can be used when programming in the Delphi development environment. This module, in COM interfaces, is designed for web developers and for those using RAD tools such as Visual Basic or Delphi



[Organization of NBioBSPCOM.dll and NSearch Engine]

### 3.6.1 Initialization / termination / parameter setting

Declare a new object for NBIOSPCOM module to start the NSearch Engine. This method initializes global variables and allocates the Memory Resident Fingerprint DB (MRFDB). Declaring the object free will free all allocated memory used in the Engine.

#### 1) Initializing the NSearch Engine object

Declare a new object to initialize the NSearch Engine.

```
objNBioBSP : variant;  
...  
objNBioBSP := CreateOleObject('NBIOSPCOM.NBioBSP');  
objNSearch := objNBioBSP.NSearch  
...
```

When the object is created, the NSearch Engine is automatically initialized. While initialization, this method internally checks license validation if expired or exceeding the maximum number of fingerprints. If the license is invalid, the initialization will fail.

Reading and updating parameters used in the NSearch Engine are only possible after initialization.

```
objNSearch.MaxCandidateNumber := 5;  
If objNSearch.ErrorCode <> 0 then  
  Begin  
    // Init module failed. Show error message.  
    // objNSearch.ErrorDescript  
  End  
Else  
  Begin  
    // Init success.  
  End;
```

#### 2) Closing NSearch Engine object

Declare the object free to terminate the NSearch Engine. This needs to be done before the application is closed.

```
objNSearch := Null;  
objNBioBSP := Null;
```

### **3) Reading NSearch Engine parameters**

The MaxCandidateNumber property can be used to read the number of candidates to be generated from the NSearch Engine.

```
ret := objNSearch.MaxCandidateNumber;
```

### **4) Setting NSearch Engine parameters**

The MaxCandidateNumber property can be specified to update the number of candidates to be generated from the NSearch Engine.

```
objNSearch.MaxCandidateNumber := 5;
```



## 3.6.2 Fingerprint Registration / Deletion / Search

The main functions of the NSearch Engine are to register, delete and search a fingerprint data. While fingerprint registration or identification, the NBioBSP FIR can be used only in text format.

### 1) Registering fingerprint

AddFIR(BSTR bszInputFIR, LONG nUserID)

This method is to register fingerprint templates into the MRFDB. The FIR can contain more than one fingerprint data for a user, thus single user registration could generate several template data into the MRFDB. Refer to the properties for more information about each template data.

```
Var
szFir      : wideString;
nUserID    : longint;
nFingerID  : longint;

// Get FIR data
objDevice := objNBioBSP.Device;
objExtraction := objNBioBSP.Extraction;

objDevice.Open(255);
objExtraction.Enroll(Null);
objDevice.Close(255);

szFir := objExtraction.TextEncodeFIR;

// Register FIR to NSearch DB
objNSearch.AddFIR (szFIR, nUserID);

For i := 0 to objNSearch.Count do
Begin
```

```
objResult := objNSearch.Item[i];  
// Print FP information in FIR.  
nFingerID := objResult.FingerID;  
End;
```

In this example code, the objResult, lcandidateList object, contains registration results for each fingerprint. The objResult also includes UserID and SampleNumber. The FingerID represents 1 through 10 as right thumb to left little finger, and the SampleNumber can be 0 or 1 as first or second. If using the Capture method, unlike the Enroll method, the SampleNumber will be always 0 value.

Each FingerID values are as follows.

```
0 : NBioAPI_FINGER_ID_UNKNOWN  
1 : NBioAPI_FINGER_ID_RIGHT_THUMB  
2 : NBioAPI_FINGER_ID_RIGHT_INDEX  
3 : NBioAPI_FINGER_ID_RIGHT_MIDDLE  
4 : NBioAPI_FINGER_ID_RIGHT_RING  
5 : NBioAPI_FINGER_ID_RIGHT_LITTLE  
6 : NBioAPI_FINGER_ID_LEFT_THUMB  
7 : NBioAPI_FINGER_ID_LEFT_INDEX  
8 : NBioAPI_FINGER_ID_LEFT_MIDDLE  
9 : NBioAPI_FINGER_ID_LEFT_RING  
10: NBioAPI_FINGER_ID_LEFT_LITTLE
```

## 2) Deleting fingerprint

### A. Deleting one template from DB

RemoveData (LONG nUserID, LONG nFingerID, LONG nSampleNumber)

RemoveData method is to delete a template from the MRFDB. This method deletes only one template. In order to delete all templates for a user, use RemoveUser method..

Note that you must also delete the corresponding data in the system database in your application to maintain data synchronization with the MRFDB.

```
nUserID := 2;
nFingerID := 1; // NBioAPI_FINGER_ID_RIGHT_THUMB
nSampleNumber := 0;
objNSearch.RemoveData(nUserID, nFingerID, nSampleNumber);
If objNSearch.ErrorCode = 0 then
  Begin
    // Success
  End
Else
  Begin
    // Fail
  End;
```

## **B. Deleting all templates for a user**

RemoveUser (LONG nUserID)

Call RemoveUser method to delete all templates for a user from the MRFDB. Use RemoveData method to delete a single template.

Note that you must also delete the corresponding data in the system database in your application to maintain data synchronization with the MRFDB.

```
nUserID := 1;
objNSearch.RemoveUser(nUserID);

If objNSearch.ErrorCode = 0 then
  Begin
    // Success
  End
Else
  Begin
```

```
// Fail  
End;
```

### 3) Search/Identification

#### A. Searching (Candidate listing)

```
SearchData(BSTR bszInputFIR)
```

```
...
```

Use the SearchData method to search the template candidates that are most closely matched with the input template in the MRFDB. Input the FIR data, then the candidate list will be returned including the UserID, FingerID, SampleNumber and ConfidenceLevel.

```
// Capture fingerprint
```

```
...
```

```
objNSearch.SearchData(szFir);
```

```
If objNSearch.ErrorCode = 0 then
```

```
Begin
```

```
For i := 0 to objNSearch.Count do
```

```
Begin
```

```
objResult := objNSearch.Item[i];
```

```
nUserID := objResult.UserID;
```

```
nFingerID := objResult.FingerID;
```

```
nSampleNumber := objResult.SampleNumber;
```

```
nConfidenceLevel := objResult.ConfidenceLevel;
```

```
End;
```

```
End;
```

#### B. Identification

```
IdentifyUser (BSTR bszInputFIR, LONG nSecuLevel)
```

Use the `IdentifyUser` method to identify a template from the MRFDB. Input the FIR data and the security level, then the identification result will be returned. If the identification is successful, the `UserID` property is returned with a value, otherwise, it remains empty.

The security level, in range from 0 to 9, determines the threshold that can be used to match a template against the other. Only if the `ConfidenceLevel` is higher than the security level setting, the matching result will be successful.

```
// Capture fingerprint
...
objNSearch.IdentifyUser (szFir, nSecuLevel);

If objNSearch.ErrorCode = 0 then
Begin
    nUserID          := objNSearch.UserID;
End;
```

### 3.6.3 MRFDB Management

When there is any change in the MRFDB, the system database must be updated with the corresponding data to maintain data synchronization with the MRFDB. In addition, since the MRFDB, allocated in the memory, will be lost when the application is closed, the MRFDB must be saved into a file if it needs to be reused when the application restarts,.

#### 1) Saving fingerprint DB

```
SaveDBToFile(BSTR bszFilePath)
```

Use the SaveDBToFile method to save the MRFDB into a file in disk. Input the full path name for the parameter.

```
szFileName := "C:\\data\\MyDB.fdb";  
objNSearch.SaveDBToFile(szFileName);
```

```
If objNSearch.ErrorCode = 0 then  
Begin  
    // Success  
End  
Else  
Begin  
    // Fail  
End;
```

#### 2) Loading fingerprint DB

```
LoadDBFromFile(BSTR bszFilePath)
```

Use the LoadDBFromFile method to load a file from disk into the MRFDB. This method can be called after the NSearch Engine restarts.

```
szFileName := "C:\\data\\MyDB.fdb";
```

```
objNSearch.LoadDBFromFile(szFileName);
```

```
If objNSearch.ErrorCode = 0 then
```

```
Begin
```

```
    // Success
```

```
End
```

```
Else
```

```
Begin
```

```
    // Fail
```

```
End;
```

### **3) Clearing fingerprint DB**

```
ClearDB()
```

To delete all template data in the MRFDB, Use the ClearDB method.

```
objNSearch.ClearDB();
```

```
If objNSearch.ErrorCode = 0 then
```

```
Begin
```

```
    // Success
```

```
End
```

```
Else
```

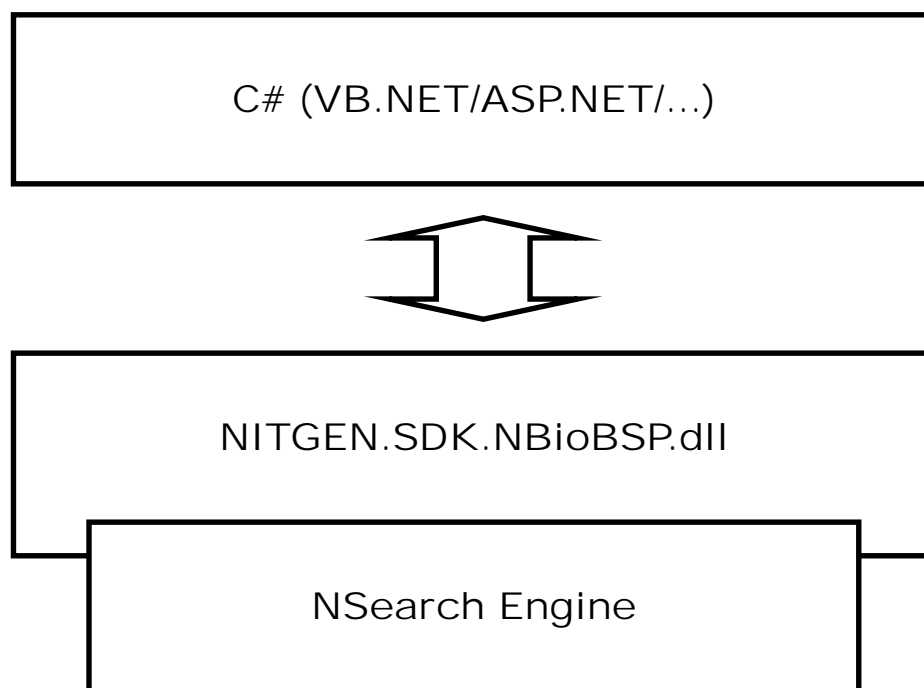
```
Begin
```

```
    // Fail
```

```
End;
```

## 3.7 C# (.NET) Programming

The NITGEN.SDK.NBioBSP.dll can be used when programming in the C# development environment. This module, in .NET Class Library interfaces, is designed for .NET developers such as C# programming language.



[ NITGEN.SDK.NBioBSP.dll and NSearch Engine Structure ]



### 3.7.1 Initialization / termination / parameter setting

Declare a new object for NITGEN.SDK.NBioBSP module to start the NSearch Engine. This method initializes global variables and allocates the Memory Resident Fingerprint DB (MRFDB). Declaring the object free will free all allocated memory used in the Engine.

#### 1) Initializing the NSearch Engine object

Declare a new object to initialize the NSearch Engine.

```
using NITGEN.SDK.NBioBSP;
...
NBioAPI m_NBioBSP = new NBioAPI();
...
NBioAPI.NSearch m_NSearch = new NBioAPI.NSearch(m_NBioBSP);
uint ret = m_NSearch.InitEngine();
if (ret == NBioAPI.Error.NONE)
    // Success
Else
    // Fail
```

When the object is created, the NSearch Engine is automatically initialized. While initialization, this method internally checks license validation if expired or exceeding the maximum number of fingerprints. If the license is invalid, the initialization will fail. Note that the license has the limitation of fingerprint that can be registered.

Use the SearchData method to search the template candidates that are most closely matched with the input template in the MRFDB

Reading and updating parameters used in the NSearch Engine are only possible after initialization. The parameter, MaxCandidateNumber, indicates the number of candidates that are most closely matched with the input template in the MRFDB. Use the SetInitInfo() method to update the value of MaxCandidateNumber. The default is 10.

```
using NITGEN.SDK.NBioBSP;
...
```

```
NBioAPI.NSearch.INIT_INFO_0 initInfo0 = new NBioAPI.NSearch.INIT_INFO_0();
initInfo0.MaxCandidateNumber = 5;
uint ret = m_NSearch.SetInitInfo(initInfo0);
if (ret != NBioAPI.Error.NONE) {
    // Init module failed. Show error message.
} else {
    // Init success.
}
```

## 2) Closing the NSearch Engine object

Use the `TerminateEngine()` method to terminate the NSearch Engine. This needs to be done before the application is closed.

```
m_NSearch.TerminateEngine();
```

## 3) Reading the NSearch Engine parameters

Use the `GetInitInfo()` method to retrieve parameters set on the NSearch Engine.

```
NBioAPI.NSearch.INIT_INFO_0 initInfo0;
ret = m_NSearch.GetInitInfo(out initInfo0);
```

## 4) Setting the NSearch Engine parameters

Use the `SetInitInfo()` method to configure parameters of the NSearch Engine.

```
ret = m_NSearch.SetInitInfo(initInfo0);
```

## 3.7.2 Fingerprint Registration / Deletion / Search

The main functions of the NSearch Engine are to register, delete and search a fingerprint data. While fingerprint registration or identification, the NBioBSP FIR can be used only in text format.

### 1) Registering fingerprint into the MRFDB

```
AddFIRToDB(...);
```

This method is to register fingerprint templates into the MRFDB (Memory Resident Fingerprint DB). The FIR can contain more than one fingerprint data for a user, thus single user registration could generate several template data into the MRFDB. Each independent template data is stored into DB, not as a set of user data. Refer to the properties for more information about each template data.

```
// Get FIR data
m_NBioBSP.OpenDevice(NBioAPI.Type.DEVICE_ID.AUTO);
uint ret = m_NBioBSP.Enroll(out hNewFIR, null);
m_NBioBSP.CloseDevice(NBioAPI.Type.DEVICE_ID.AUTO);

// Regist FIR to NSearch DB
NBioAPI.NSearch.FP_INFO[] fpInfo;
m_NSearch.AddFIR(hNewFIR, nUserID, out fpInfo);

// Add item to list of SearchDB
foreach (NBioAPI.NSearch.FP_INFO sampleInfo in fpInfo)
{
    // Print FP information in FIR
    ListViewItem listItem = new ListViewItem();
    listItem.Text = sampleInfo.ID.ToString();
    listItem.SubItems.Add(sampleInfo.FingerID.ToString());
    listItem.SubItems.Add(sampleInfo.SampleNumber.ToString());
    listSearchDB.Items.Add(listItem);
}
```

In this example code, the parameter, `fplInfo`, contains registration results for each fingerprint. The `fplInfo` also includes `UserID`, `FingerID` and `SampleNumber`. The `FingerID` represents 1 through 10 as right thumb to left little finger, and the `SampleNumber` can be 0 or 1 as first or second. If using the `Capture` method, unlike the `Enroll` method, the `SampleNumber` will be always 0 value.

Each `FingerID` values are as follows.

```
0 : NBioAPI_FINGER_ID_UNKNOWN
1 : NBioAPI_FINGER_ID_RIGHT_THUMB
2 : NBioAPI_FINGER_ID_RIGHT_INDEX
3 : NBioAPI_FINGER_ID_RIGHT_MIDDLE
4 : NBioAPI_FINGER_ID_RIGHT_RING
5 : NBioAPI_FINGER_ID_RIGHT_LITTLE
6 : NBioAPI_FINGER_ID_LEFT_THUMB
7 : NBioAPI_FINGER_ID_LEFT_INDEX
8 : NBioAPI_FINGER_ID_LEFT_MIDDLE
9 : NBioAPI_FINGER_ID_LEFT_RING
10: NBioAPI_FINGER_ID_LEFT_LITTLE
```

## 2) Deleting fingerprint

### A. Deleting one template from the MRFDB

```
RemoveData(...);
```

`RemoveData` method is to delete a template from the MRFDB. This method deletes only one template. In order to delete all templates for a user, use `RemoveUser` method..

Note that you must also delete the corresponding data in the system database in your application to maintain data synchronization with the MRFDB.

```
nUserID = 2;
```

```
nFingerID = 1;
nSampleNumber = 0;
ret = m_NSearch.RemoveData(nUserID, nFingerID, nSampleNumber);
if (ret != NBioAPI.Error.NONE)
    // Success
else
    // Fail
```

## **B. Deleting all templates for a user from the MRFDB**

```
RemoveUser (uint nUserID);
```

Call RemoveUser method to delete all templates for a user from the MRFDB. Use RemoveData method to delete a single template.

Note that you must also delete the corresponding data in the system database in your application to maintain data synchronization with the MRFDB.

```
uint nUserID = 1
uint ret = m_NSearch.RemoveUser(nUserID);

if (ret != NBioAPI.Error.NONE)
    // Success
else
    // Fail
```

## **3) Search/Identification**

### **A. Search (Candidate listing)**

```
SearchData (...);
```

Use the SearchData method to search the template candidates that are most closely matched with the input template in the MRFDB. Input the FIR data, then the candidate

list will be returned as NBioAPI.NSearch.CANDIDATE including the UserID, FingerID, SampleNumber and ConfidenceLevel. The higher the ConfidenceLevel is, the more possibility the result is correct. The candidates are in descending order.

```
// Search FIR to NSearch DB
NBioAPI.NSearch.CANDIDATE[] candidate;
ret = m_NSearch.SearchData(hCapturedFIR, out candidate);
if (ret == NBioAPI.Error.NONE)
{
    // Add item to list of result
    foreach (NBioAPI.NSearch.CANDIDATE candInfo in candidate)
    {
        ListViewItem listItem = new ListViewItem();
        listItem.Text = candInfo.ID.ToString();
        listItem.SubItems.Add(candInfo.FingerID.ToString());
        listItem.SubItems.Add(candInfo.SampleNumber.ToString());
        listItem.SubItems.Add(candInfo.ConfidenceLevel.ToString());
        listResult.Items.Add(listItem);
    }
}
```

## B. Identification

IdentifyData (...);

Use the IdentifyData method to identify a template from the MRFDB. Input the FIR data and the security level, then the identification result will be returned. If the identification is successful, the NBioAPI.NSearch.FP\_INFO structure is returned with a value, otherwise, it remains empty.

The security level, in range from 0 to 9, determines the threshold that can be used to match a template against the other. Only if the ConfidenceLevel is higher than the security level setting, the matching result will be successful.

```
// Identify FIR to NSearch DB
```

```
NBioAPI.NSearch.FP_INFO fpInfo;  
uint ret = m_NSearch.IdentifyData(  
    hCapturedFIR,  
    NBioAPI.Type.FIR_SECURITY_LEVEL.NORMAL,  
    out fpInfo);  
if (ret == NBioAPI.Error.NONE)  
{  
    nUserID = fpInfo.ID;  
}
```

### 3.7.3 MRFDB Management

When there is any change in the MRFDB, the system database must be updated with the corresponding data to maintain data synchronization with the MRFDB. In addition, since the MRFDB, allocated in the memory, will be lost when the application is closed, the MRFDB must be saved into a file if it needs to be reused when the application restarts,.

#### 1) Saving the MRFDB into a file

```
SaveDBToFile(string szFilePath)
```

Use the SaveDBToFile method to save the MRFDB into a file in disk. Input the full path name for the parameter.

```
string szFileName = "C:\\data\\MyDB.fdb";
uint ret = m_NSearch.SaveDBToFile(szFileName);
if (ret != NBioAPI.Error.NONE)
    // Success
else
    // Fail
```

#### 2) Loading the MRFDB from a file

```
LoadDBFromFile(string szFilePath)
```

Use the LoadDBFromFile method to load a file from disk into the MRFDB. This method can be called after the NSearch Engine restarts.

```
string szFileName = "C:\\data\\MyDB.fdb"
uint ret = m_NSearch.LoadDBFromFile(szFileName);
if (ret != NBioAPI.Error.NONE)
    // Success
else
    // Fail
```



### 3) Clearing the MRFDB

```
ClearDB();
```

Use the ClearDB method to delete all template stored in the MRFDB.

```
uint ret = m_NSearch.ClearDB();  
if (ret != NBioAPI.Error.NONE)  
    // Success  
else  
    // Fail
```

# Appendix A. API Reference

This chapter describes more about the data structures, error code, constant and APIs used in the NSearch Engine. The header file, 'NBioAPI\_NSearchType.h', includes the data structures, error codes and constant values, and all NSearch Engine APIs are defined in the file 'NBioAPI\_NSearch.h'.

## A.1 Functions

### 1) NSearch Engine Initialization / Termination

**NBioAPI\_InitNSearchEngine** : Initializes the NSearch Engine.

```
NBioAPI_RETURN NBioAPI_InitNSearchEngine(  
    IN          NBioAPI_HANDLE          hHandle  
);
```

#### Description

This function initializes the NSearch Engine. It allocates the memory for the Memory Resident Fingerprint DB (MRFDB) and initializes the global variables.

#### Parameters

hHandle : Handle of the NBioBSP module.

#### Return Values

NBioAPIERROR\_NONE : Success.

NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.

NBioAPIERROR\_NSEARCH\_OPEN\_FAIL :

Failed to open the NSearch.dll file.

NBioAPIERROR\_NSEARCH\_INIT\_FAIL :

Failed to initialize the NSearch Engine.

NBioAPIERROR\_NSEARCH\_MEM\_OVERFLOW :

Failed to allocate the memory

NBioAPIERROR\_NSEARCH\_LICENSE\_LOAD :

Failed to load the license file.

NBioAPIERROR\_NSEARCH\_LICENSE\_KEY :

Incorrect key value in the license file.

NBioAPIERROR\_NSEARCH\_LICENSE\_EXPIRED :

The license has been expired.

**NBioAPI\_TerminateNSearchEngine** : Closes the NSearch Engine.

```
NBioAPI_RETURN NBioAPI_TerminateNSearchEngine (  
    IN          NBioAPI_HANDLE          hHandle  
);
```

### **Description**

This function is to close the NSearch Engine. It must be called before the application is closed to free all memory allocated used in the Engine.

### **Parameters**

hHandle : Handle of the NBioBSP module.

### **Return Values**

NBioAPIERROR\_NONE : Success.

NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP의 Handle is used.

NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

**NBioAPI\_GetNSearchInitInfo** : Reads the NSearch Engine parameters.

```
NBioAPI_RETURN NBioAPI_GetNSearchInitInfo(  
    IN          NBioAPI_HANDLE          hHandle  
    IN          NBioAPI_UINT8          nStructureType,  
    OUT         NBioAPI_INIT_INFO_PTR   pInitInfo  
);
```

### Description

This function reads the parameter values used in the NSearch Engine. The StructType must be 0 on the current version.

### Parameters

hHandle : Handle of the NBioBSP module.  
nStructureType : Type of the NBioAPI\_INIT\_INFO structure. (must be 0).  
pInitInfo : Pointer of the NBioAPI\_INIT\_INFO structure.

### Return Values

NBioAPIERROR\_NONE : Success.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP Handle is used.  
NBioAPIERROR\_INVALID\_POINTER : Invalid pointer is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.  
NBioAPIERROR\_STRUCTTYPE\_NOT\_MATCHED :  
Invalid NBioAPI\_INIT\_INFO structure for the StructType.

**NBioAPI\_SetNSearchInitInfo** : Sets the NSearch Engine parameters

```
NBioAPI_RETURN NBioAPI_SetNSearchInitInfo(  
    IN          NBioAPI_HANDLE          hHandle  
    IN          NBioAPI_UINT8          nStructureType,  
    IN          NBioAPI_INIT_INFO_PTR   pInitInfo  
);
```

### Description

This function is used to set the NSearch Engine parameters. The StructType must be 0 on the current version.

### Parameters

hHandle : Handle of the NBioBSP module.  
nStructureType : Type of the NBioAPI\_INIT\_INFO structure. (must be 0).  
pInitInfo : Pointer of the NBioAPI\_INIT\_INFO structure.

### Return Values

NBioAPIERROR\_NONE : Successful.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.  
NBioAPIERROR\_INVALID\_POINTER : Invalid pointer is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.  
NBioAPIERROR\_STRUCTTYPE\_NOT\_MATCHED :  
Invalid NBioAPI\_INIT\_INFO structure for the StructType.  
NBioAPIERROR\_INIT\_MAXCANDIDATE : Invalid MaxCandidate value is set.

## 2) Fingerprint Registration / Deletion / Search

**NBioAPI\_AddFIRToNSearchDB** : Registers fingerprint data.

```
NBioAPI_RETURN NBioAPI_AddFIRToNSearchDB(  
    IN          NBioAPI_HANDLE          hHandle  
    IN const    NBioAPI_INPUT_FIR_PTR    pInputFIR,  
    IN          NBioAPI_UINT32          nUserID,  
    OUT         NBioAPI_NSEARCH_SAMPLE_INFO_PTR  pSampleInfo  
);
```

### Description

This function is used to register fingerprint templates into the MRFDB. The template information will be returned after successful registration.

### Parameters

**hHandle** : Handle of the NBioBSP module.  
**pInputFIR** : FIR data. (Refer to the NBioAPI manual)  
**nUserID** : user ID.  
**pSampleInfo** : Detailed information about the templates registered.

### Return Values

**NBioAPIERROR\_NONE** : Successful.  
**NBioAPIERROR\_INVALID\_HANDLE** : Invalid NBioBSP handle is used.  
**NBioAPIERROR\_INVALID\_POINTER** : Invalid pointer is used.  
**NBioAPIERROR\_NSEARCH\_INIT\_FAIL** : The Engine has not been initialized.  
**NBioAPIERROR\_NSEARCH\_OVER\_LIMIT** :  
Exceeding the number of fingerprints specified in the license.  
**NBioAPIERROR\_NSEARCH\_INVALID\_TEMPLATE** :  
Invalid template is entered.  
**NBioAPIERROR\_NSEARCH\_MEM\_OVERFLOW** :  
Failed to allocate the memory.

**NBioAPI\_RemoveDataFromNSearchDB** : Deletes a template from the DB.

```
NBioAPI_RETURN NBioAPI_RemoveDataFromNSearchDB(  
    IN          NBioAPI_HANDLE          hHandle  
    IN          NBioAPI_NSEARCH_FP_INFO_PTR  pFpInfo  
);
```

### Description

This function is used to delete a template from the MRFDB. Input the pointer of NBioAPI\_NSEARCH\_FP\_INFO structure including the user ID, finger number and sample number, then the corresponding template will be deleted.

### Parameters

hHandle : Handle of the NBioBSP module.  
pFpInfo : Pointer of NBioAPI\_NSEARCH\_FP\_INFO structure.

### Return Values

NBioAPIERROR\_NONE : Successful.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.  
NBioAPIERROR\_INVALID\_POINTER : Invalid pointer is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.



**NBioAPI\_RemoveUserFromNSearchDB** : Deletes all templates of a user.

```
NBioAPI_RETURN NBioAPI_RemoveUserFromNSearchDB(  
    IN          NBioAPI_HANDLE          hHandle  
    IN          NBioAPI_UINT32          nUserID  
);
```

### **Description**

This function is used to delete all templates of a user from the MRFDB.

### **Parameters**

hHandle : Handle of the NBioBSP module.  
nUserID : user ID.

### **Return Values**

NBioAPIERROR\_NONE : Successful.

NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.

NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

**NBioAPI\_SearchDataFromNSearchDB** : Searches and returns candidates.

```
NBioAPI_RETURN NBioAPI_SearchDataFromNSearchDB(  
    IN          NBioAPI_HANDLE          hHandle  
    IN const    NBioAPI_INPUT_FIR_PTR    pInputFIR,  
    OUT         NBioAPI_NSEARCH_CANDIDATE_PTR* ppCandidate,  
    IN          NBioAPI_VOID_PTR         pReserved = NULL  
);
```

### Description

This function is used to search the template candidates that are most closely matched with the input template in the MRFDB. The candidate list will be returned including the FingerNumber, SampleNumber, and ConfidenceLevel.

Calling NBioAPI\_FreeNSearchCandidate() function will free the memory allocated for the candidate list.

### Parameters

hHandle : Handle of the NBioBSP module.  
pInputFIR : FIR data. (Refer to the NBioAPI manual)  
ppCandidate : Pointer of NBioAPI\_NSEARCH\_CANDIDATE structure.  
pReserved : must be NULL.

### Return Values

NBioAPIERROR\_NONE : Successful.

NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.

NBioAPIERROR\_INVALID\_POINTER : Invalid pointer is used.

NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

NBioAPIERROR\_NSEARCH\_INVALID\_TEMPLATE :  
Invalid template is entered.

**NBioAPI\_FreeNSearchCandidate** : Frees the memory for candidate list.

```
NBioAPI_RETURN NBioAPI_FreeNSearchCandidate(  
    IN          NBioAPI_HANDLE          hHandle  
    IN          NBioAPI_NSEARCH_CANDIDATE_PTR pCandidate  
);
```

### **Description**

This function frees the memory allocated for the candidate list from the NBioAPI\_SearchDataFromNSearchDB() function.

### **Parameters**

hHandle : Handle of the NBioBSP module.  
pCandidate : Pointer of NBioAPI\_NSEARCH\_CANDIDATE structure.

### **Return Values**

NBioAPIERROR\_NONE : Successful.

NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.

**NBioAPI\_IdentifyDataFromNSearchDB** : Identified a template.

```
NBioAPI_RETURN NBioAPI_IdentifyDataFromNSearchDB(  
    IN          NBioAPI_HANDLE          hHandle  
    IN const    NBioAPI_INPUT_FIR_PTR    pInputFIR,  
    IN          NBioAPI_FIR_SECURITY_LEVEL nSecuLevel,  
    OUT         NBioAPI_NSEARCH_FP_INFO_PTR pFpInfo  
);
```

### Description

This function is to identify a template from the MRFDB. Input the FIR and the security level, then the identification result will be returned. If the identification is successful, the pFpInfo parameter in the NBioAPI\_NSEARCH\_FP\_INFO structure is filled up with fingerprint information matched, otherwise, it remains empty. The fingerprint information contains ID, FingerID and SampleNumber. The security level, in range from 0 to 9, determines the threshold that can be used to match a template against the other. Only if the ConfidenceLevel is higher than the security level setting, the matching result will be successful.

### Parameters

hHandle : Handle of the NBioBSP module.  
pInputFIR : FIR data. (Refer to the NBioAPI manual)  
nSecuLevel : Security level (0~9).  
pFpInfo : Fingerprint information of the identification result

### Return Values

NBioAPIERROR\_NONE : Successful.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.  
NBioAPIERROR\_NSEARCH\_IDENTIFY\_FAIL :  
  Cannot find the matched template.  
NBioAPIERROR\_NSEARCH\_INVALID\_TEMPLATE :  
  Invalid template is entered.

### 3) DB Management

**NBioAPI\_SaveNSearchDBToFile** : Saves the MRFDB into a file in disk.

```
NBioAPI_RETURN NBioAPI_SaveNSearchDBToFile(  
    IN          NBioAPI_HANDLE          hHandle  
    IN const    NBioAPI_CHAR*           szFilepath  
);
```

#### Description

This function is used to save the MRFDB into a file in disk. Input the full path name for the szFilePath parameter.

#### Parameters

hHandle : Handle of the NBioBSP module.  
szFilePath : Full path to save the MRFDB into a file in disk.

#### Return Values

NBioAPIERROR\_NONE : Successful.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.  
NBioAPIERROR\_NSEARCH\_SAVE\_DB : Failed to save the DB.

**NBioAPI\_LoadNSearchDBFromFile** : Loads the file into the MRFDB.

```
NBioAPI_RETURN NBioAPI_LoadNSearchDBFromFile(  
    IN          NBioAPI_HANDLE          hHandle  
    IN const    NBioAPI_CHAR*           szFilepath  
);
```

### Description

This function is to load the file into the MRFDB.

### Parameters

hHandle : Handle of the NBioBSP module.  
szFilename : Full path to load into the MRFDB.

### Return Values

NBioAPIERROR\_NONE : Successful.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.  
NBioAPIERROR\_NSEARCH\_LOAD\_DB : Failed to load the DB.

**NBioAPI\_ImportIndexSearchDBToNSearchDB** : Loads the IndexSearch backup file into the MRFDB.

```
NBioAPI_RETURN NBioAPI_ImportIndexSearchDBToNSearchDB (  
    IN          NBioAPI_HANDLE          hHandle  
    IN const    NBioAPI_CHAR*          szFilepath  
);
```

### Description

This function is to load the IndexSearch backup file into the MRFDB.

### Parameters

hHandle : Handle of the NBioBSP module.  
szFilename : Fullpath to load into the MRFDB.

### Return Values

NBioAPIERROR\_NONE : Successful.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.  
NBioAPIERROR\_INDEXSEARCH\_LOAD\_DB : Failed to load the DB.  
NBioAPIERROR\_INDEXSEARCH\_UNKOWN\_VER : Invalid version.

**NBioAPI\_ClearNSearchDB** : Clears the MRFDB.

```
NBioAPI_RETURN NBioAPI_ClearNSearchDB(  
    IN          NBioAPI_HANDLE          hHandle  
);
```

### **Description**

This function is used to delete all templates from the MRFDB.

### **Parameters**

hHandle : Handle of the NBioBSP module.

### **Return Values**

NBioAPIERROR\_NONE : Successful.

NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.

NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.



**NBioAPI\_GetDataCountFromNSearchDB** : Retrieves the count of fingerprint DB.

```
NBioAPI_RETURN NBioAPI_GetDataCountNSearchDB (  
    IN          NBioAPI_HANDLE          hHandle,  
    OUT         NBioAPI_UINT32*         pDataCount  
);
```

### Description

This function is to retrieve the count of fingerprint data from the MRFDB.

### Parameters

hHandle : Handle of the NBioBSP module.  
pDataCount : Count of fingerprint data.

### Return Values

NBioAPIERROR\_NONE : Successful.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.  
NBioAPIERROR\_INVALID\_POINTER : Invalid pointer is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

**NBioAPI\_CheckDataExistFromNSearchDB** : Checks the existence of a fingerprint.

```
NBioAPI_RETURN NBioAPI_CheckDataExistFromNSearchDB(  
    IN          NBioAPI_HANDLE          hHandle  
    IN          NBioAPI_NSEARCH_FP_INFO_PTR  pFpInfo,  
    OUT         NBioAPI_BOOL*           pExist  
);
```

### Description

This function is used to check the existence of a specific fingerprint data in the MRFDB.

### Parameters

hHandle : Handle of the NBioBSP module.  
pFpInfo : Fingerprint information to check its existence.  
pExist : The result of checking existence.

### Return Values

NBioAPIERROR\_NONE : Successful.  
NBioAPIERROR\_INVALID\_HANDLE : Invalid NBioBSP handle is used.  
NBioAPIERROR\_INVALID\_POINTER : Invalid pointer is used.  
NBioAPIERROR\_NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

## A.2 NSearch Engine Structure

### NBioAPI\_NSEARCH\_INIT\_INFO\_0

```
typedef struct NBioAPI_NSearch_init_info_0 {
    NBioAPI_UINT32    StructureType;           /* must be 0 */
    NBioAPI_UINT32    MaxCandidateNumber;     /* Default = 10 */
    NBioAPI_UINT32    researved0;             /* Reserved */
    NBioAPI_UINT32    researved1;             /* Reserved */
    NBioAPI_UINT32    researved2;             /* Reserved */
    NBioAPI_UINT32    researved3;             /* Reserved */
    NBioAPI_UINT32    researved4;             /* Reserved */
    NBioAPI_UINT32    researved5;             /* Reserved */
    NBioAPI_UINT32*    researved6;            /* Reserved */
} NBioAPI_NSEARCH_INIT_INFO_0,
  NBioAPI_NSEARCH_INIT_INFO_PTR_0;
```

#### Description

Initial parameters for the NSearch Engine.

#### Members

StructureType : Type of NBioAPI\_INIT\_INFO structure. (must be 0).  
MaxCandidateNumber : Number of candidates to be returned.  
researved0 ~ researved6 : Reserved

### NBioAPI\_NSEARCH\_FP\_INFO

```
typedef struct NBioAPI_NSearch_fp_info {
    NBioAPI_UINT32    ID;
    NBioAPI_UINT8     FingerID; /* NBioAPI_FINGER_ID */
    NBioAPI_UINT8     SampleNumber;
} NBioAPI_NSEARCH_FP_INFO, *NBioAPI_NSEARCH_FP_INFO_PTR;
```

#### Description

Fingerprint information of users.

### Members

ID; : User ID.

FingerID : Finger IDs as follows.

0	: NBioAPI_FINGER_ID_UNKNOWN
1	: NBioAPI_FINGER_ID_RIGHT_THUMB
2	: NBioAPI_FINGER_ID_RIGHT_INDEX
3	: NBioAPI_FINGER_ID_RIGHT_MIDDLE
4	: NBioAPI_FINGER_ID_RIGHT_RING
5	: NBioAPI_FINGER_ID_RIGHT_LITTLE
6	: NBioAPI_FINGER_ID_LEFT_THUMB
7	: NBioAPI_FINGER_ID_LEFT_INDEX
8	: NBioAPI_FINGER_ID_LEFT_MIDDLE
9	: NBioAPI_FINGER_ID_LEFT_RING
10	: NBioAPI_FINGER_ID_LEFT_LITTLE

SampleNumber : Sample number. (first or second)

### NBioAPI\_NSEARCH\_CANDIDATE

```
typedef struct NBioAPI_NSearch_candidate {
    NBioAPI_UINT32      ID;
    NBioAPI_UINT8      FingerID; /* NBioAPI_FINGER_ID */
    NBioAPI_UINT8      SampleNumber;
    NBioAPI_UINT8      ConfidenceLevel;
} NBioAPI_NSEARCH_CANDIDATE,
    *NBioAPI_NSEARCH_CANDIDATE_PTR;
```

### Description

Template information of the candidate list.

### Members

ID; : User ID.

FingerID : Finger ID.

SampleNumber: Sample number.

ConfidenceLevel : Confidence level (0~9). Only if the ConfidenceLevel is higher than the security level setting, the matching result will be successful.

## **NBioAPI\_NSEARCH\_SAMPLE\_INFO**

```
typedef struct NBioAPI_NSearch_sample_info {  
    NBioAPI_UINT32          ID;  
    NBioAPI_UINT8          SampleCount[11];  
} NBioAPI_NSEARCH_SAMPLE_INFO,  
  *NBioAPI_NSEARCH__SAMPLE_INFO_PTR;
```

### **Description**

Fingerprint information of registration.

### **Members**

ID; : User ID.

SampleCount : Sample ID of each finger.

## A.3 Error Codes & Constants

### 1) Error codes

Error code	Definition
NBioAPIERROR_NONE	No error.
NBioAPIERROR_INVALID_HANDLE	Invalid NBioBSP Handle.
NBioAPIERROR_INVALID_POINTER	Invalid pointer.
NBioAPIERROR_STRUCTTYPE_NOT_MATCHED	Invalid type for the NBioAPI_INIT_INFO structure.
NBioAPIERROR_INIT_MAXCANDIDATE	Invalid MaxCandidate value.
NBioAPIERROR_NSEARCH_OPEN_FAIL	Failed to open the NSearch.dll.
NBioAPIERROR_NSEARCH_INIT_FAIL	Engine not initialized.
NBioAPIERROR_NSEARCH_MEM_OVERFLOW	Failed to allocate memory.
NBioAPIERROR_NSEARCH_SAVE_DB	Failed to save the MRFDB.
NBioAPIERROR_NSEARCH_LOAD_DB	Failed to load the MRFDB.
NBioAPIERROR_NSEARCH_INVALID_TEMPLATE	Invalid template data.
NBioAPIERROR_NSEARCH_OVER_LIMIT	Exceeding the number of fingerprints in the license.
NBioAPIERROR_NSEARCH_IDENTIFY_FAIL	Identification failed.
NBioAPIERROR_NSEARCH_LICENSE_LOAD	Failed to load license file.
NBioAPIERROR_NSEARCH_LICENSE_KEY	License key not matched.
NBioAPIERROR_NSEARCH_LICENSE_EXPIRED	License expired.

### 2) Constants

NBioAPI_CONF_LEVEL_LOWEST	1
NBioAPI_CONF_LEVEL_LOWER	2
NBioAPI_CONF_LEVEL_LOW	3

---

NBioAPI_CONF_LEVEL_BELOW_NORMAL	4
NBioAPI_CONF_LEVEL_NORMAL	5
NBioAPI_CONF_LEVEL_ABOVE_NORMAL	6
NBioAPI_CONF_LEVEL_HIGH	7
NBioAPI_CONF_LEVEL_HIGHER	8
NBioAPI_CONF_LEVEL_HIGHEST	9

# Appendix B. COM Reference

The NBIOBSPCOM.dll can be used when programming in the Visual Basic or Delphi development environment. This module, in COM interfaces, is designed for web developers and for those using RAD such as Visual Basic or Delphi

This chapter describes the Search function only on the NBioBSP COM module.

(Refer to the NBioBSP SDK manual)

## B.1 Properties ( Search Object )

### **long    ErrorCode**

Records the error code used in the last method or property. If successful, it returns zero, otherwise, returns the error code.

### **BSTR    ErrorDescription**

Records the description of the error code in text type.

### **long    Count**

Indicates the number of fingerprints returned.

### **long    MaxCandidatenum**

Indicates the maximum number of candidates that can be returned.

### **long    GetDataCountFromDB**

Returns the number of templates in the Memory Resident Fingerprint DB (MRFDB).

### **BOOL    CheckDataExistFromDB**

**(long nUserID, long nFingerID, long nSampleNumber)**

Checks if the template exists in the MRFDB.

### **long    UserID**

Indicates the user ID that is returned after identification.



## B.2 Properties ( CandidateList Object )

The following properties are the object that can be used to display the matching result or registration result. Implemented in Collection, these objects can be used without declaration.

### **long    UserID**

Indicates the user ID that must be in number.

### **long    FingerID**

Indicates the finger ID.

NBioAPI_FINGER_ID_UNKNOWN	= 0
NBioAPI_FINGER_ID_RIGHT_THUMB	= 1
NBioAPI_FINGER_ID_RIGHT_INDEX	= 2
NBioAPI_FINGER_ID_RIGHT_MIDDLE	= 3
NBioAPI_FINGER_ID_RIGHT_RING	= 4
NBioAPI_FINGER_ID_RIGHT_LITTLE	= 5
NBioAPI_FINGER_ID_LEFT_THUMB	= 6
NBioAPI_FINGER_ID_LEFT_INDEX	= 7
NBioAPI_FINGER_ID_LEFT_MIDDLE	= 8
NBioAPI_FINGER_ID_LEFT_RING	= 9
NBioAPI_FINGER_ID_LEFT_LITTLE	= 10

### **long    SampleNumber**

Indicates the sample ID of each fingerprint. ( 0 / 1)

### **long    ConfidenceLevel**

Returns the matching score after fingerprint searching. (1 – 9, 9 is the highest)

## B.3 Methods

**AddFIR (VARIANT FIR, long nUserID)**

**Description**

This method is used to register the templates for a user.

**Parameter**

FIR : Fingerprint data created by the NBioBSP module.

nUserID : User ID for the FIR. It must be in number.

**Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

CandidateList Object : Returns the result of registration. The ConfidenceLevel is not used.

**RemoveData(long nUserID, long nFingerID, long nSampleNumber)****Description**

This method is used to delete a template from the MRFDB.

**Parameter**

nUserID : User ID to be deleted. It must be in number.

nFingerID : Finger ID to be deleted.

nSampleNumber : Sample ID of the finger to be deleted. It can be 0 or 1.

**Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

**RemoveUser(long nUserID)****Description**

This method is used to delete all templates of a user.

**Parameter**

nUserID : User ID to be deleted. It must be in number.

**Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

**SearchData(VARIANT storedFIR)****Description**

This method is to search a template in the MRFDB and returns a candidate list.

**Parameter**

storedFIR : The FIR data to be searched.

**Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

CandidateList Object : The result list of searching.

**IdentifyUser(VARIANT storedFIR, long nSecuLevel)****Description**

This method is to identify a template in the MRFDB and return the result.

**Parameter**

storedFIR : The FIR data to be identified.

nSecuLevel : Security level.

NBioAPI_FIR_SECURITY_LEVEL_LOWEST	= 1
NBioAPI_FIR_SECURITY_LEVEL_LOWER	= 2
NBioAPI_FIR_SECURITY_LEVEL_LOW	= 3
NBioAPI_FIR_SECURITY_LEVEL_BELOW_NORMAL	= 4

NBioAPI_FIR_SECURITY_LEVEL_NORMAL	= 5
NBioAPI_FIR_SECURITY_LEVEL_ABOVE_NORMAL	= 6
NBioAPI_FIR_SECURITY_LEVEL_HIGH	= 7
NBioAPI_FIR_SECURITY_LEVEL_HIGHER	= 8
NBioAPI_FIR_SECURITY_LEVEL_HIGHEST	= 9

#### **Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

UserID : User ID identified.

#### **Clear()**

##### **Description**

This method is to delete all data in the MRFDB.

##### **Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

#### **SaveDBToFile(BSTR bszFilePath)**

##### **Description**

This method is to save the MRFDB into a file in disk.

##### **Parameter**

bszFilePath : Full path name to save the MRFDB.

##### **Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

## **LoadDBFromFile(BSTR bszFilePath)**

### **Description**

This method is to load the file from disk into the MRFDB.

### **Parameter**

bszFilePath : Full path name to load the FDB file.

### **Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

## **ImportIndexSearchDB(BSTR bszFilePath)**

### **Description**

This method is to load the IndexSearch backup file from disk into the MRFDB.

### **Parameter**

bszFilePath : Full path name to load the IndexSearch FDB file.

### **Relation Property**

ErrorCode : Returns the result. If successful, returns NBioAPIERROR\_NONE.

ErrorDescript : Returns the description of the error code.

# Appendix C. .NET Reference

## C.1 NSearch Engine Methods

### 1) NSearch Engine Initialization / Termination

**InitEngine** : Initializes the NSearch Engine.

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

#### Description

This function initializes the NSearch Engine. It allocates the memory for the Memory Resident Fingerprint DB (MRFDB) and initializes the global variables.

#### Parameters

N/A

#### Return Values

NBioAPI.Error.NONE : Success.

NBioAPI.Error.NSEARCH\_OPEN\_FAIL :

Failed to open the NSearch.dll file.

NBioAPI.Error.NSEARCH\_INIT\_FAIL :

Failed to initialize the NSearch Engine.

NBioAPI.Error.NSEARCH\_MEM\_OVERFLOW :

Failed to allocate the memory

NBioAPI.Error.NSEARCH\_LICENSE\_LOAD :

Failed to load the license file.

NBioAPI.Error.NSEARCH\_LICENSE\_KEY :

Incorrect key value in the license file.

NBioAPI.Error.NSEARCH\_LICENSE\_EXPIRED :

The license has been expired.

**TerminateEngine** : Closes the NSearch Engine.

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

### **Description**

This function is to close the NSearch Engine. It must be called before the application is closed to free all memory allocated used in the Engine.

### **Parameters**

N/A

### **Return Values**

NBioAPI.Error.NONE : Success.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

**GetInitInfo** : Reads the NSearch Engine parameters.

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

### **Description**

This function reads the parameter values used in the NSearch Engine. The StructType must be 0 on the current version.

### **Parameters**

InitInfo0 : Pointer of the NBioAPI\_INIT\_INFO structure.

### **Return Values**

NBioAPI.Error.NONE : Success.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.



**SetInitInfo** : Sets the NSearch Engine parameters

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

### **Description**

This function is used to set the NSearch Engine parameters. The StructType must be 0 on the current version.

### **Parameters**

InitInfo0 : Pointer of the NBioAPI\_INIT\_INFO structure.

### **Return Values**

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

NBioAPI.Error.INIT\_MAXCANDIDATE : Invalid MaxCandidate value is set.

## 2) Fingerprint Registration / Deletion / Search

**AddFIR** : Registers fingerprint data.

```
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_TEXTENCOD 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 );
```

### Description

This function is used to register fingerprint templates into the MRFDB. The template information will be returned after successful registration.

### Parameters

CapturedFIR : FIR data. (Refer to the NBioAPI manual)  
 UserID : user ID.  
 SampleInfo : Detailed information about the templates registered.

### Return Values

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

NBioAPI.Error.NSEARCH\_OVER\_LIMIT :

Exceeding the number of fingerprints specified in the license.

NBioAPI.Error.NSEARCH\_INVALID\_TEMPLATE: Invalid template is entered.

NBioAPI.Error.NSEARCH\_MEM\_OVERFLOW : Failed to allocate the memory.

**RemoveData** : Deletes a template from the DB.

```
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 );
```

### Description

This function is used to delete a template from the MRFDB. Input the NBioAPI.NSearch.FP\_INFO structure including the user ID, finger ID and sample number, then the corresponding template will be deleted.

### Parameters

FpInfo : Template Information (UserID / FingerID / SampleNumber)  
UserID : User ID  
FingerID : Finger ID  
SampleNumber : Sample number

### Return Values

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

**RemoveUser** : Deletes all templates of a user.

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

### **Description**

This function is used to delete all templates of a user from the MRFDB.

### **Parameters**

nUserID : User ID.

### **Return Values**

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

**SearchData** : Searches and returns candidates.

```
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)
```

### Description

This function is used to search the template candidates that are most closely matched with the input template in the MRFDB. The candidate list will be returned including the UserID, FingerID, SampleNumber, and ConfidenceLevel.

### Parameters

CapturedFIR : FIR data. (Refer to the NBioAPI manual)  
Candidate : Candidate list.

### Return Values

NBioAPI.Error.NONE : Successful.  
NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.  
NBioAPI.Error.NSEARCH\_INVALID\_TEMPLATE : Invalid template is entered.

**IdentifyData** : Identified a template.

```
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_TEXTENCOD 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 );
```

## Description

This function is to identify a template from the MRFDB. Input the FIR and the security level, then the identification result will be returned. If the identification is successful, the pFpInfo parameter in the NBioAPI\_NSEARCH\_FP\_INFO structure is filled up with fingerprint information matched, otherwise, it remains empty. The fingerprint information contains ID, FingerID and SampleNumber. The security level, in range from 0 to 9, determines the threshold that can be used to match a template against the other. Only if the ConfidenceLevel is higher than the security level setting, the matching result will be successful.

## Parameters

CapturedFIR : FIR data. (Refer to the NBioAPI manual)

SecuLevel : Security level (0~9).

FpInfo : Fingerprint information of the identification result

### **Return Values**

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_IDENTIFY\_FAIL : Cannot find the matched template.

NBioAPI.Error.NSEARCH\_INVALID\_TEMPLATE : Invalid template is entered.



### 3) DB Management

**SaveDBToFile** : Saves the MRFDB into a file in disk.

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

#### Description

This function is used to save the MRFDB into a file in disk. Input the full path name for the szFilePath parameter.

#### Parameters

szFilepath : Full path to save the MRFDB into a file in disk.

#### Return Values

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

NBioAPI.Error.NSEARCH\_SAVE\_DB : Failed to save the DB.

**LoadDBFromFile** : Loads the file into the MRFDB.

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

### **Description**

This function is to load the file into the MRFDB.

### **Parameters**

szFilepath : Full path to load into the MRFDB.

### **Return Values**

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

NBioAPI.Error.NSEARCH\_LOAD\_DB : Failed to load the DB.

**ImportIndexSearchDBToNSearchDB** : Loads the IndexSearch backup file into the MRFDB.

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

### **Description**

This function is to load the IndexSearch backup file into the MRFDB.

### **Parameters**

szFilepath : Full path to load into the MRFDB.

### **Return Values**

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

NBioAPI.Error.INDEXSEARCH\_LOAD\_DB : Failed to load the DB.

NBioAPI.Error.INDEXSEARCH\_UNKOWN\_VER : Invalid version.

**ClearDB** : Clears the MRFDB.

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

### **Description**

This function is used to delete all templates from the MRFDB.

### **Parameters**

N/A

### **Return Values**

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

**GetDataCount** : Retrieves the count of fingerprint DB.

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

### **Description**

This function is to retrieve the count of fingerprint data from the MRFDB.

### **Parameters**

DataCount : Count of fingerprint data.

### **Return Values**

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

**CheckDataExist** : Checks the existence of a fingerprint.

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

### Description

This function is used to check the existence of a specific fingerprint data in the MRFDB.

### Parameters

FpInfo : Fingerprint information to check its existence.  
Exist : The result of checking existence.

### Return Values

NBioAPI.Error.NONE : Successful.

NBioAPI.Error.NSEARCH\_INIT\_FAIL : The Engine has not been initialized.

## C.2 NSearch Engine Structure

### NBioAPI.NSearch.INIT\_INFO\_0

```
public struct NBioAPI.NSearch.INIT_INFO_0 {  
    System.UInt32      StructureType;           /* must be 0 */  
    System.UInt32      MaxCandidateNumber;      /* Default = 10 */  
    System.UInt32      Researved0;              /* Reserved */  
    System.UInt32      Researved1;              /* Reserved */  
    System.UInt32      Researved2;              /* Reserved */  
    System.UInt32      Researved3;              /* Reserved */  
    System.UInt32      Researved4;              /* Reserved */  
    System.UInt32      Researved5;              /* Reserved */  
    System.UInt32*     Researved6;              /* Reserved */  
}
```

### Description

Initial parameters for the NSearch Engine.

### Members

StructureType : Type of NBioAPI.NSearch.INIT\_INFO structure. (must be 0).  
MaxCandidateNumber : Number of candidates to be returned.  
researved0 ~ researved6 : Reserved

### NBioAPI.NSearch.FP\_INFO

```
public struct NBioAPI.NSearch.FP_INFO{  
    System.UInt32      ID;  
    System.Byte        FingerID; /* NBioAPI_FINGER_ID */  
    System.Byte        SampleNumber;  
}
```

### Description

Fingerprint information of users.

### Members

ID; : User ID.

FingerID : Finger Identifier number.  
SampleNumber : Sample number. (first or second)

#### **NBioAPI.NSearch.CANDIDATE**

```
public struct NBioAPI.NSearch.CANDIDATE{  
    System.UInt32      ID;  
    System.Byte        FingerID; /* NBioAPI_FINGER_ID */  
    System.Byte        SampleNumber;  
    System.Byte        ConfidenceLevel;  
}
```

#### **Description**

Template information of the candidate list.

#### **Members**

ID : user ID.  
FingerID : Finger ID.  
SampleNumber : Sample number.  
ConfidenceLevel : Confidence level (0~9). Only if the ConfidenceLevel is higher than the security level setting, the matching result will be successful.

#### **NBioAPI.NSearch.SAMPLE\_INFO**

```
public struct NBioAPI.NSearch.SAMPLE_INFO {  
    System.UInt32      ID;  
    System.Byte        SampleCount[11];  
}
```

#### **Description**

Fingerprint information of registration.

#### **Members**

ID : User ID.  
SampleCount : Sample ID of each finger.