

FaceCell 1.2 EDK

FaceCell 1.2 EDK

Published September 7, 20010. Version 1.2.0.1
Copyright © 2007-2010 Neurotechnology

Table of Contents

1. About	1
1.1. Introduction	1
1.2. Platforms Supported	1
1.3. System Requirements	1
1.4. Licensing	1
2. What's New	2
2.1. Version 1.2.0.0	2
2.2. Version 1.1.0.0	2
2.3. Version 1.0.1.0	2
2.4. Version 1.0.0.2	2
2.5. Version 1.0.0.1	2
2.6. Version 1.0.0.0	2
3. Overview	3
3.1. Image Support	3
3.1.1. Image	3
3.1.2. Image Format	3
3.1.3. Low-Level Image Input-Output	4
3.2. Matching threshold and similarity	4
3.3. Face Image Constraints	4
3.3.1. Face image constraints	5
4. Tutorials	6
4.1. About	6
4.2. C Tutorials	6
4.2.1. Enroll face from an image	6
4.2.2. Enroll face from a stream	7
4.2.3. Find all the faces in an image	8
4.2.4. Find eyes of a face	8
4.2.5. Identify a face from the image	9
4.2.6. Verify a face from the image with a template	9
5. Samples	11
5.1. Windows CE	11
5.1.1. Pocket PC 2003	11
5.1.2. Windows Mobile 5.0	11
6. Reference (C/C++)	12
6.1. FccExtractor Library	12
6.1.1. FccExtractor Module	12
6.2. FccMatcher Library	27
6.2.1. FccMatcher Module	27
6.3. NCore Library	35
6.3.1. NCore Module	36
6.3.2. NErrors Module	36
6.3.3. NGeometry Module	37
6.3.4. NLibraryInfo Module	40
6.3.5. NMemory Module	42
6.3.6. NParameters Module	43
6.3.7. NTypes Module	44
6.4. NImages Library	50
6.4.1. Bmp Module	50
6.4.2. Jpeg Module	55
6.4.3. NGrayscaleImage Module	58
6.4.4. NImageFormat Module	60
6.4.5. NImage Module	70
6.4.6. NImages Module	84
6.4.7. NMonochromeImage Module	85
6.4.8. NPixelFormat Module	87
6.4.9. NRgbImage Module	89
6.4.10. Png Module	91
6.4.11. Tiff Module	96
6.5. NTemplate Library	97
6.5.1. NTemplate Module	98
6.5.2. NFRecord Module	114

6.5.3. NFTemplate Module	186
6.5.4. NLRecord Module	203
6.5.5. NLTemplate Module	211
6.5.6. NETemplate Module	227
6.5.7. NERRecord Module	243
A. Support	257
B. Distribution Content	258
B.1. bin	258
B.2. documentation	258
B.3. include	258
B.4. lib	258
B.5. samples	258
B.6. tutorial	259
C. Change Log	260
C.1. Components	260
C.1.1. FccExtractor Library	260
C.1.2. FccMatcher Library	261
C.1.3. NCore Library	261
C.1.4. NImages Library	263
C.1.5. NTemplate Library	264
C.2. Samples	265
C.2.1. Windows	265

List of Figures

6.1. Structure of NTemplate.	98
-----------------------------------	----

Chapter 1. About

1.1. Introduction

Before using FaceCell EDK components protection service must be started.

- ARM Linux users should start `/bin/linux_arm/activation/run_pgd.sh`
- MS Windows CE 2003 users should start `\bin\ppc03_armv4\activation\pg.exe`
- MS Windows CE 2003 or later users should start `\bin\ppc05_armv4i\activation\pg.exe`

Note for Windows Mobile users: Start program `pg.exe` and close its window, it will stay in background, this must be done each time you reboot your Windows Mobile)

FaceCell EDK consists of [FccExtractor](#), [FccMatcher](#) main libraries. [NCore](#) and [NImages](#) are auxiliary libraries that provides infrastructure and functionality for working with images. The [samples programs](#) were developed to show how to use FaceCell EDK components.

1.2. Platforms Supported

FaceCell EDK supports platforms based on ARM processor architecture. Libraries for Windows Mobile and ARM Linux operating systems are provided.

1.3. System Requirements

- ARM based 400 MHz processor is recommended for face enrollment in less than two seconds. Supported ARM processor core families are: XScale, StrongArm, ARM11, ARM10, ARM9.
- At least 8 Mb of memory for FaceCell code and data arrays.
- Embedded camera with at least 320 x 240 pixels physical resolution (640 x 480 pixels recommended).
- Linux or MS Windows Mobile 2003 (or later).

Please note, that FaceCell source code EDK could be easily ported to most other platforms and processors.

1.4. Licensing

Neurotechnology grants you a personal, non-exclusive license to use the Software for 30 consecutive days on one embedded device for the purpose of technology evaluation.

Chapter 2. What's New

2.1. Version 1.2.0.0

- Improved face detection.

2.2. Version 1.1.0.0

- Improved face and eyes detection precision.
- Improved enrollment from sequence of images (generates more reliable template).
- Added quality measure to facial features extraction. It can be controlled by face quality threshold.
- Added NLRecord library that is required by FccExtractor and FccMatcher libraries.

2.3. Version 1.0.1.0

- Improved eye detection.
- Updated components to newer versions.

2.4. Version 1.0.0.2

- Updated components to newer versions.

2.5. Version 1.0.0.1

- JPEG image format support was added.

2.6. Version 1.0.0.0

- Initial release.

Chapter 3. Overview

3.1. Image Support

Image support in the FaceCell EDK can be divided into the following three parts:

- [Image](#). The base of all image support. Developers should start using this part and take advantage of other parts if it is required.
- [Image Format](#). Declares the supported image formats. Shows how to load and save images in a format-neutral way.
- [Low-Level Image Input-Output](#). Should be used to have more control on how images are loaded and saved in particular format.

3.1.1. Image

Image is a rectangular area of pixels (image elements), defined by width, height and pixel format.

Pixel format describes type of color information contained in the image like monochrome, grayscale, true color or palette-based (indexed) and describes pixels storage in memory (how many bits are required to store one pixel).

Image in the FaceCell EDK is defined by [HNIImage](#) handle in [NImage](#) module. It is an encapsulation of a memory block that stores image pixels. The memory block is organized as rows that follow each other in top-to-bottom order. The number of rows is equal to height of image. Each row is organized as pixels that follow each other in left-to-right order. The number of pixels in a row is equal to width of image. A pixel format describes how image pixels are stored. See [NImageGetWidth](#), [NImageGetHeight](#), [NImageGetStride](#), [NImageGetPixelFormat](#) and [NImageGetPixels](#) functions for more information.

An image can have horizontal and vertical resolution attributes assigned to it if they are applicable (they are required for image, and do not make sense for face image). See [NImageGetHorzResolution](#) and [NImageGetVertResolution](#) functions for more information.

An image can be created either as empty or from existing memory block. See [NImageCreate](#), [NImageCreateFromData](#) and [NImageCreateWrapper](#) functions for more information.

For each value of [NPixelFormat](#) exposed via interface a module is provided for managing according type of image (getting and setting individual pixels, etc.). See [NGrayscaleImage](#), [NMonochromeImage](#) and [NRGBImage](#) modules for more information.

An image can be converted to different pixel format using [NImageCreateFromImage](#) or [NImageCreateFromImageEx](#) function.

Different methods should be used to display an image on different platforms:

- On Windows [BmpSaveImageToHBitmap](#) function can be used to receive a standard Win32 HBITMAP for the image. The reverse process is also possible using [BmpLoadImageFromHBitmap](#) function.
- On Linux there is no easy method implemented. However, a memory block containing pixels of image could be accessed via [NImageGetPixelFormat](#) function. The memory block can be used to display the image or convert it to some other representation on any platform.

An image can be stored in file in any supported [image format](#) using [NImageSaveToFile](#) function.

An image stored in file in any supported [image format](#) can be loaded using [NImageCreateFromFile](#) function.

3.1.2. Image Format

Image format is a specification of [image](#) storage in a file. The specification may require to compress/decompress image during writing/reading it to/from a file.

Image format in the FaceCell EDK is defined by [HNIImageFormat](#) handle in [NImageFormat](#) module.

There is a number of image formats supported in the FaceCell EDK. Certain formats could not be read from and written to a file on all platforms. See the following table for details.

Image Format	Can read	Can write
BMP	Yes	Yes
GIF	No	No
JPEG	Yes	Yes
PNG	Yes	Yes
TIFF	Yes	No

These image formats are accessible using [NImageFormatGetBmp](#) and [NImageFormatGetTiff](#) functions.

To find out which images formats are supported in the FaceCell EDK in version-independent way [NImageFormatGetFormatCount](#) and [NImageFormatGetFormat](#) functions should be used.

Name, file name pattern (file filter) and default file extension of the image format can be retrieved using [NImageFormatGetName](#), [NImageFormatGetFileFilter](#) and [NImageFormatGetDefaultFileExtension](#) functions.

To find out which image format should be used to read or write a particular file [NImageFormatSelect](#) function should be used.

An image can be loaded and saved from/to file or memory buffer using [NImageFormatLoadImageFromFile](#), [NImageFormatLoadImageFromMemory](#), [NImageFormatSaveImageToFile](#) and [NImageFormatSaveImageToMemory](#) functions. Note that not all image formats support both reading and writing. Use [NImageFormatCanRead](#) and/or [NImageFormatCanWrite](#) function(s) to check if the particular image format does.

3.1.3. Low-Level Image Input-Output

Low-level image I/O in the FaceCell EDK is implemented in [Bmp](#) and [Tiff](#) modules.

These modules provides functions for loading and saving [images](#) in according format (BMP and TIFF).

Those functions can take parameters that precisely control loading and saving of the image in particular formats.

3.2. Matching threshold and similarity

FaceCell features matching algorithm provides value of features collections similarity as a result. The higher is similarity, the higher is probability that features collections are obtained from the same person.

Matching threshold is linked to false acceptance rate (FAR, different subjects erroneously accepted as of the same) of matching algorithm. The higher is threshold, the lower is FAR and higher FRR (false rejection rate, same subjects erroneously accepted as different) and vice a versa.

FAR	Threshold
1%	0.625
0.1%	0.650
0.01%	0.675
0.001%	0.700
0.0001%	0.725
0.00001%	0.750

3.3. Face Image Constraints

3.3.1. Face image constraints

Face recognition is very sensitive to image quality so maximum care should be attributed to image acquisition.

3.3.1.1. Pose

The frontal pose (full-face) must be used. Rotation of the head must be less than ± 5 degrees from frontal in every direction - nodded up/down, rotated left/right, tilted right/left.

3.3.1.2. Expression

The expression should be neutral (non-smiling) with both eyes open, and mouth closed. Every effort should be made to have supplied images comply with this specification. A smile with closed jaw is allowed but not recommended.

3.3.1.2.1. Examples of Non-Recommended Expressions

- A smile where the inside of the mouth is exposed (jaw open).
- Raised eyebrows.
- Closed eyes.
- Eyes looking away from the camera.
- Squinting.
- Frowning.
- Hair covering eyes.
- Rim of glasses covering part of the eye.

3.3.1.3. Face changes

Beard, moustache and other changeable face features influence face recognition quality and if frequent face changes are typical for some individual, face database should contain e.g. face with beard and cleanly shaved face enrolled with identical ID.

3.3.1.4. Lighting

Lighting must be equally distributed on each side of the face and from top to bottom. There should be no significant direction of the light or visible shadows. Care must be taken to avoid "hot spots". These artifacts are typically caused when one, high intensity, focused light source is used for illumination.

3.3.1.5. Eyeglasses

There should be no lighting artifacts on eyeglasses. This can typically be achieved by increasing the angle between the lighting, subject and camera to 45 degrees or more. If lighting reflections cannot be removed, then the glasses themselves should be removed. (However this is not recommended as face recognition typically works best when matching people with eyeglasses against themselves wearing the same eyeglasses).

Glasses have to be of clear glass and transparent so the eyes and irises are clearly visible. Heavily tinted glasses are not acceptable.

3.3.1.6. Web cameras

Embedded camera with at least 320 x 240 pixels physical resolution (640 x 480 pixels recommended).

Images should be enrolled and matched using the same camera, as devices have different optical distortions that can influence face recognition performance.

Chapter 4. Tutorials

4.1. About

This chapter of documentation describes and in depth comments tutorial programs sources provided in the FaceCell EDK. It provides you with basic overview of workflow when performing fundamental tasks. Tutorial source code is available for the following programming languages:

1. [C tutorials](#)

4.2. C Tutorials

1. [Enroll face from an image](#),
2. [Enroll face from a stream](#),
3. [Find all the faces in an image](#),
4. [Find eyes of a face](#),
5. [Verify a face from the image with a template](#),
6. [Identify a face from the image](#).

4.2.1. Enroll face from an image

This tutorial describes the process of extracting face features from an image using the C programming language. The source for this tutorial can be found in `/tutorials/C/EnrollImage.c`.

1. FaceCell EDK is made of several parts each performing different subset of whole SDK functions. Before writing any program the dependencies for a particular task must be determined and set.

```
#include <NCore.h>
#include <NImages.h>
#include <FccExtractor.h>
```

2. Next step is to load an image from which face features will be extracted.

```
NResult result;
HNImage image;
char *fileName;
fileName = /* obtain file name */
result = NImageCreateFromFile(fileName, NULL, &image);
```

3. FaceCell EDK works only with grayscale images. So images must be converted to grayscale before being used with the FaceCell EDK functions:

```
HNImage grayscale;
result = NImageCreateFromImage(npfGrayscale, 0, image, &grayscale);
```

4. All the parameters of extraction routines are stored in an extractor object, so it must be initialized before usage.

```
HFccExtractor extractor;
result = FcceCreate(&extractor);
```

5. Before extracting face features buffer space has to be allocated for these features to be stored. The size of this buffer can be determined this way:

```
NSizeType maxSize;
result = FcceGetMaxTemplateSize(extractor, &maxSize);
```

6. The simplest way to extract face feature templates from an image is to use function that will automatically detect face in an image and return face features template and face detection results.

```
void *template;
FcceDetectionDetails details;
template = (void *) malloc(maxSize);
result = FcceExtract(extractor, grayscale, &details, template, maxSize, &size);
```

7. After extraction the details of the face detection can be accessed and checked this way:

```
if (details.FaceAvailable)
{
    printf("found face:\n");
    printf("\tlocation = (%d, %d), width = %d, height = %d,
           confidence = %.2f\n",
           details.Face.Rectangle.X, details.Face.Rectangle.Y,
           details.Face.Rectangle.Width, details.Face.Rectangle.Height,
           details.Face.Confidence);
}
if (details.EyesAvailable)
{
    printf("\tfound eyes:\n");
    printf("\t\tfirst: location = (%d, %d), confidence = %.2f\n",
           details.Eyes.First.X, details.Eyes.First.Y,
           details.Eyes.FirstConfidence);
    printf("\t\tsecond: location = (%d, %d), confidence = %.2f\n",
           details.Eyes.Second.X, details.Eyes.Second.Y,
           details.Eyes.SecondConfidence);
}
```

8. After the image has been used and isn't necessary anymore it must be freed:

```
NImageFree(grayscale);
```

9. When the extractor is not necessary, it must be freed:

```
FcceFree(extractor);
```

4.2.2. Enroll face from a stream

This tutorial describes the process of extracting face features from an image stream using the C programming language. The source for this tutorial can be found in `/tutorials/C/EnrollStream.c`.

1. For [includes](#), [image loading](#), [converting to grayscale](#) and [extractor initialization](#), [maximum template size](#) see "Enrolling face from an image" tutorial.
2. Before enrolling a person using image stream extractor parameters must be initialized. Currently there is only one parameter to be set - the number of frames from which face features should be extracted. This initialization must be done every time a new persons face features are to be extracted.

```
Int numberOfFrames = /* get the number of frames */
result = FcceExtractStart(extractor, numberOfFrames);
```

3. After initializing enrollement from stream the frames must be processed in frame by frame fashion, checking the results after each processing step.

```
NSizeType size;
NBool stopped;
stopped = NFalse;
size = 0;
for (i = 0; !stopped && (i < numberOfFrames); ++i)
{
    result = FcceExtractNext(extractor, images[i], &details,
                             template, maxSize, &size, &stopped);
}
```

4. For [detection results](#), [image](#) and [extractor freeing](#) see "Enrolling face from an image" tutorial.

4.2.3. Find all the faces in an image

This tutorial describes the process of finding all the face regions (bounding boxes) in an image using the C programming language. The source for this tutorial can be found in `/tutorials/C/FindFaces.c`.

1. For [includes](#), [image loading](#), [converting to grayscale](#) and [extractor initialization](#), [maximum template size](#) see "Enrolling face from an image" tutorial.
2. FaceCell EDK has several functions related to face and eyes finding. First of all all face region have to be found in an image:

```
int faceCount;
FcceFace *faces;
result = FcceDetectFaces(extractor, grayscale, &faceCount, &faces);
```

3. In FaceCell EDK the C function that detects all the faces allocates buffer space for the found faces structures, but this buffer isn't freed automatically, so it must be freed manually:

```
Nfree(faces);
```

4. For [detection results](#), [image](#) and [extractor freeing](#) see "Enrolling face from an image" tutorial.

4.2.4. Find eyes of a face

This tutorial describes the process of finding face eyes in an image using the C programming language. The source for this tutorial can be found in `/tutorials/C/FindEyes.c`.

1. For [includes](#), [image loading](#), [converting to grayscale](#) and [extractor initialization](#), [maximum template size](#) see "Enrolling face from an image" tutorial.
2. For face region [detection](#) and [remarks](#) see "Finding all the faces in an image" tutorial.
3. After all the face regions have been found in the image, these region can be further searched for face eyes:

```
FcceEyes *eyes;
eyes = new FcceEyes[faceCount];
for (int i = 0; i < faceCount; ++i)
{
    result = FcceDetectEyes(extractor, grayscale, &faces[i], &eyes);
}
```

4. In FaceCell EDK the C function that detects all the faces allocates buffer space for the found faces structures, but

this buffer isn't freed automatically, so it must be freed manually:

```
Nfree(faces);
```

5. For [detection results](#), [image](#) and [extractor freeing](#) see "Enrolling face from an image" tutorial.

4.2.5. Identify a face from the image

This tutorial describes the process of identifying face features against a database of collected face features using the C programming language. The source for this tutorial can be found in `/tutorials/C/Identify.c`.

1. For [includes](#), [image loading](#), [converting to grayscale](#) and [extractor initialization](#), [maximum template size](#), [extracting face features from an image](#) see "Enrolling face from an image" tutorial.
2. All the parameters of matching routines are stored in a matcher object, so it must be initialized before usage:

```
HFccMatcher matcher;  
result = FccmCreate(&matcher);
```

3. FaceCell EDK has several functions for template matching. If one template should be checked against all the database it is advised to use identification functions. To start identification procedure, you must first initialize matcher for this task by telling which template will be matched against all the database.

```
result = FccmIdentifyStart(matcher, template, size);
```

4. After the matcher has been initialized the templates from the database should be matched one by one with the template that was used for initialization.

```
Int templateCount = /* obtain template count */  
void **templates;  
Int *sizes;  
templates = (void **) malloc(sizeofTemplate * templateCount);  
/* Obtain templates */  
for (i = 0; i < templateCount; ++i)  
{  
    result = FccmIdentifyNext(matcher, templates[i], sizes[i], &score);  
}
```

5. After template identification is finished, the memory, taken by the identification routines must be freed:

```
result = FccmIdentifyEnd(matcher);
```

6. When the matcher is not necessary, it must be destroyed:

```
FccmFree(matcher);
```

7. For [detection results](#), [image](#) and [extractor freeing](#) see "Enrolling face from an image" tutorial.

4.2.6. Verify a face from the image with a template

This tutorial describes the process of verifying face features with another face features using the C programming language. The source for this tutorial can be found in `/tutorials/C/Verify.c`.

1. For [includes](#), [image loading](#), [converting to grayscale](#) and [extractor initialization](#), [maximum template size](#), [extracting face features from an image](#) see "Enrolling face from an image" tutorial. [Matcher initialization](#) can be found in "Identifying a template" tutorial.

2. When just two templates have to be matched against each other, verification function of FaceCell EDK should be used:

```
NDouble score;  
result = FccmVerify(matcher, template, size, enrolledTemplate,  
    enrolledSize, &score);
```

3. For [matcher freeing](#) see "[Identifying a template](#)" tutorial.
4. For [detection results](#), [image](#) and [extractor freeing](#) see "[Enrolling face from an image](#)" tutorial.

Chapter 5. Samples

Before running samples read [Face Image Constraints](#) chapter.

5.1. Windows CE

5.1.1. Pocket PC 2003

This sample application demonstrates the use of FaceCell EDK functions for simple tasks: enrolling face from an image and identifying it against a database. This sample application works only with pictures (no camera support).

5.1.2. Windows Mobile 5.0

This sample application demonstrates the use of FaceCell EDK functions for simple tasks: enrolling face from an image or a camera and identifying it against a database.

Chapter 6. Reference (C/C++)

Libraries

FccExtractor	FccExtractor is a library for finding faces in images and extracting their features into templates.
FccMatcher	Provides functionality to verify and match VeriLook face feature templates.
NCore	Provides infrastructure for Neurotechnology components. NErrors defines error codes used in Neurotechnology components.
NImages	Provides functionality for loading, saving and converting images in various formats.
NTemplate	Provides functionality for packing, unpacking and editing Neurotechnology Templates (NTemplates), Fingers Templates (NFTemplates), Faces Templates (NLTemplates), Iris Templates (NETemplates), Finger Records (NFRecords), Face Records (NLRecords), Iris Records (NERecords).

6.1. FccExtractor Library

FccExtractor is a library for finding faces in images and extracting their features into templates.

Windows

Import library: `FccExtractor.dll.lib`.

DLL: `FccExtractor.dll`.

Requirements:

- [NCore.dll](#).
- [NImage.dll](#).

Linux

Shared object: `libFccExtractor.so`.

Requirements:

- [libNCore.so](#).
- [libNImages.so](#).

Modules

FccExtractor	FccExtractor is a part of FccExtractor library intended for human face finding in pictures and extracting their features into VeriLook face feature templates.
------------------------------	--

6.1.1. FccExtractor Module

FccExtractor is a part of FccExtractor library intended for human face finding in pictures and extracting their features into VeriLook face feature templates.

Header file: FccExtractor.h (includes NCore.h, NImages.h, NGeometry.h and FccExtractor-Params.h)

Functions

FcceCopyParameters	Copies parameter values from one FccExtractor to another.
FcceCreate	Creates a FccExtractor.
FcceDetectEyes	Finds eyes of a given face region.
FcceDetectFace	Finds one face region that best fits extraction requirements from the given image.
FcceDetectFaces	Finds all face regions in the given image.
FcceExtract	Fully automatically extracts feature template from the face that best fits extraction requirements from the given image.
FcceExtractNext	Extracts one features template from image sequence
FcceExtractStart	Initializes features extraction routines that uses image sequences to extract features.
FcceExtractUsingEyes	Extracts feature template from the face with given eye coordinates from an image.
FcceFree	Deletes the FccExtractor. After the object is deleted the specified handle is no longer valid.
FcceGeneralize	Generates generalized template from a set of templates.
FcceGetMaxTemplateSize	Retrieves maximal size of feature template that the specified FccExtractor can extract.
FcceGetParameter	Retrieves value of the specified parameter of the specified FccExtractor.
FcceIsRegistered	Checks if FccExtractor library is registered.
FcceReset	Sets default values for all parameters of the specified FccExtractor.
FcceSetParameter	Sets value of the specified parameter of the specified FccExtractor.

Types

HFccExtractor	Handle to FccExtractor object.
---------------	--------------------------------

Structures

FcceDetectionDetails	Structure with information about face detection results in a face detection routine.
FcceEyes	Structure defining information of an eye pair.
FcceFace	Structure describing face region information

Macros

FCCEP_FACE_CONFIDENCE_THRESHOLD	Identifier of type N_TYPE_DOUBLE .
FCCEP_COPYRIGHT	Identifier specifying library copyright static read-only parameter of type N_TYPE_STRING .
FCCEP_GENERALIZATION_THRESHOLD	Identifier of type N_TYPE_DOUBLE . Has the same meaning for features generalization as FC-CMP_MATCHING_THRESHOLD parameter for features matching.
FCCEP_MAX_IOD	Identifier of type N_TYPE_INT specifying maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection routines.
FCCEP_MIN_IOD	Identifier of type N_TYPE_INT specifying minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection routines.
FCCEP_NAME	Identifier specifying library name static read-only parameter of type N_TYPE_STRING .
FCCEP_VERSION_HIGH	Identifier specifying high part of library version static read-only parameter of type N_TYPE_UINT . Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
FCCEP_VERSION_LOW	Identifier specifying low part of library version static read-only parameter of type N_TYPE_UINT . Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

See Also

[FccExtractor Library](#)

6.1.1.1. FcceEyes structure

Structure defining information of an eye pair.

```
typedef struct FcceEyes_ { } FcceEyes;
```

Fields

<i>First</i>	First eye coordinates.
<i>Second</i>	Second eye coordinates.
<i>FirstConfidence</i>	How confidently the first eye was found.
<i>SecondConfidence</i>	How confidently the second eye was found.

See Also

[FccMatcher Module](#)

6.1.1.1.1. FcceEyes.First Field

First eye coordinates.

```
NPoint First;
```

See Also

[FcceEyes](#)

6.1.1.1.2. FcceEyes.Second Field

Second eye coordinates.

```
NPoint Second;
```

See Also

[FcceEyes](#)

6.1.1.1.3. FcceEyes.FirstConfidence Field

How confidently the first eye was found.

```
NDouble FirstConfidence;
```

See Also

[FcceEyes](#)

6.1.1.1.4. FcceEyes.SecondConfidence Field

How confidently the second eye was found.

```
NDouble SecondConfidence;
```

See Also

[FcceEyes](#)

6.1.1.2. FcceFace structure

Structure describing face region information

```
typedef struct FcceFace_ { } FcceFace;
```

Fields

Rectangle	Structure with face rectangle information.
Confidence	How confidently the face region was found.

See Also

[FccMatcher Module](#)

6.1.1.2.1. FcceFace.rectangle Field

Structure with face rectangle information.

```
NRect Rectangle;
```

See Also[FcceFace](#)**6.1.1.2.2. FcceFace.confidence Field**

How confidently the face region was found.

```
NDouble Confidence;
```

See Also[FcceFace](#)**6.1.1.3. FcceDetectionDetails structure**

Structure with information about face detection results in a face detection routine.

```
typedef struct FcceDetectionDetails_ { } FcceDetectionDetails;
```

Fields

FaceAvailable	Variable showing if a face region was found in an image.
Face	Structure with face region information if face was found.
EyesAvailable	Variable showing if eyes were found in the face region pointed by the <i>face</i> .
Eyes	Structure with found eyes information.

See Also[FcceMatcher Module](#)**6.1.1.3.1. FcceDetectionDetails.FaceAvailable Field**

Variable showing if a face region was found in an image.

```
NBool FaceAvailable;
```

See Also[FcceDetectionDetails](#)**6.1.1.3.2. FcceDetectionDetails.Face Field**

Structure with face region information if face was found.

```
FcceFace Face;
```

See Also[FcceDetectionDetails](#)**6.1.1.3.3. FcceDetectionDetails.EyesAvailable Field**

Variable showing if eyes were found in the face region pointed by the *face*.

```
NBool EyesAvailable;
```

See Also[FcceDetectionDetails](#)**6.1.1.3.4. FcceDetectionDetails.Eyes Field**

Structure with found eyes information.

```
FcceEyes Eyes;
```

See Also[FcceDetectionDetails](#)**6.1.1.4. FcceCopyParameters Function**

Copies parameter values from one FccExtractor to another.

```
NResult N_API FcceCopyParameters(
    HFccExtractor hDstExtractor,
    HFccExtractor hSrcExtractor
);
```

Parameters

<i>hDstExtractor</i>	[in] Handle to the destination FccExtractor object.
<i>hSrcExtractor</i>	[in] Handle to the source FccExtractor object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hDstExtractor</i> or <i>hSrcExtractor</i> is NULL .

See Also[FcceExtractor Module](#) | [HFccExtractor](#)**6.1.1.5. FcceCreate Function**

Creates a FccExtractor.

```
NResult N_API FcceCreate(
    HFccExtractor * pHExtractor
);
```

Parameters

<i>pHExtractor</i>	[out] Pointer to HFccExtractor that receives handle to created FccExtractor object.
--------------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHExtractor</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [FcceFree](#) function.

See Also

[FccExtractor Module](#) | [HFccExtractor](#) | [FcceFree](#)

6.1.1.6. FcceDetectEyes Function

Finds eyes of a given face region.

```
NResult N_API FcceDetectEyes(
    HFccExtractor hExtractor,
    HNIImage hImage,
    FcceFace *pFace,
    FcceEyes *pEyes
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
<i>hImage</i>	[in] Handle to the source image.
<i>pFace</i>	[in] Pointer to the FcceFace structure of a face found in the image <i>hImage</i> .
<i>pEyes</i>	[out] Pointer to the FcceEyes structure of eyes pair found in the face region <i>pFace</i> .

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> or <i>hImage</i> or <i>pFace</i> or <i>pEyes</i> is NULL .

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.7. FcceDetecFace Function

Finds one face region that best fits extraction requirements from the given image.


```

NResult N_API FcceDetectFace(
    HFccExtractor hExtractor,
    HNImage hImage,
    NBool *pDetected,
    FcceFace *pFace
);

```

Parameters

<i>hExtractor</i>	[in] Handle of the FccExtractor object.
<i>hImage</i>	[in] Handle of the source image.
<i>pDetected</i>	[out] Pointer to NBool variable that is set to true if face is found in the image else it's false
<i>pFace</i>	[out] Pointer to the FcceFace structure of found face.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> or <i>hImage</i> or <i>pDetected</i> or <i>pFace</i> is NULL .

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.8. FcceDetectFaces Function

Finds all face regions in the given image.

```

NResult N_API FcceDetectFaces(
    HFccExtractor hExtractor,
    HNImage hImage,
    NInt *pFaceCount,
    FcceFace **pArFaces
);

```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
<i>hImage</i>	[in] Handle to the source image.
<i>pFaceCount</i>	[out] Pointer to the NInt variable that is set to the number of found faces in the image.
<i>pArFaces</i>	[out] pointer to an array of FcceFace structures of found faces.

Remarks

The buffer for the `pArFaces` variable is allocated inside this function automatically, but you must free this variable yourself when it becomes unnecessary for you. It must be freed using NCore library function [NFree](#).

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> or <i>hImage</i> or <i>pFaceCount</i> or <i>pArFaces</i> is NULL .

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.9. FcceExtract Function

Fully automatically extracts feature template from the face that best fits extraction requirements from the given image.

```
NResult N_API FcceExtract(
    HFccExtractor hExtractor,
    HImage hImage,
    FcceDetectionDetails pDetectionDetails,
    void *pBuffer,
    NSizeType bufferSize,
    NSizeType *pSize
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
<i>hImage</i>	[in] Handle to the source image.
<i>pDetectionDetails</i>	[out] Pointer to the FcceDetectionDetails structure of face detection results in the image
<i>pBuffer</i>	[out] Address to the memory space allocated for the extracted features.
<i>bufferSize</i>	[in] Size of the allocated buffer for the extracted features.
<i>pSize</i>	[out] Returns number of bytes filled in the buffer.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> or <i>hImage</i> or <i>pDetectionDetails</i> or <i>pBuffer</i> or <i>pSize</i> is NULL .

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.10. FcceExtractNext Function

Extracts one features template from image sequence

```
NResult N_API FcceExtractNext(
    HFccExtractor hExtractor,
    HImage hImage,
    FcceDetectionDetails pDetectionDetails,
    void *pBuffer,
    NSizeType bufferSize,
    NSizeType *pSize,
    NBool *pStopped,
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
<i>hImage</i>	[in] Handle to the source image.
<i>pDetectionDetails</i>	[out] Pointer to the FcceDetectionDetails structure of face detection results in the image
<i>pBuffer</i>	[out] address to the memory space allocated for the extracted features.
<i>bufferSize</i>	[in] Size of the allocated buffer for the extracted features.
<i>pSize</i>	[out] Returns number of bytes filled in the buffer.
<i>stopped</i>	[out] Indicates the finish of feature extraction from image set.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> or <i>hImage</i> or <i>pBuffer</i> or <i>pSize</i> or <i>pStopped</i> is NULL .
N_E_INVALID_OPERATION	FcceExtractStart() was not called before FcceExtractNext() .

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.11. FcceExtractStart Function

Initializes features extraction routines that uses image sequences to extract features.

```
NResult N_API FcceExtractStart(
    HFccExtractor hExtractor,
    NInt durationInFrames
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
<i>durationInFrames</i>	[in] Maximum number of frames which will be used for one feature template extraction.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>durationInFrames</i> is smaller than 1.

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.12. FcceExtractUsingEyes Function

Extracts feature template from the face with given eye coordinates from an image.

```
NResult N_API FcceExtractUsingEyes(
    HFccExtractor hExtractor,
    HNImage hImage,
    FcceEyes *pEyes,
    void *pBuffer,
    NSizeType bufferSize,
    NSizeType *pSize
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractsor object.
<i>hImage</i>	[in] Handle to the source image.
<i>pEyes</i>	[in] Pointer to FcceEyes structure of the face eye co-ordinates in the image.
<i>pBuffer</i>	[out] Address to the memory space allocated for the Extracted features.
<i>bufferSize</i>	[in] Size of the allocated buffer for the Extracted features.
<i>pSize</i>	[out] Returns number of bytes filled in the buffer.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> or <i>hImage</i> or <i>pBuffer</i> or <i>pSize</i> or <i>pEyes</i> is NULL .

See Also

[FccExtractUsingEyesor Module](#) | [HFccExtractUsingEyesor](#)

6.1.1.13. FcceFree Function

Deletes the FccExtractor. After the object is deleted the specified handle is no longer valid.

```
void N\_API FcceFree(
    HFccExtractor hExtractor
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
-------------------	---

Remarks

If *hExtractor* is [NULL](#), does nothing.

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.14. FcceGeneralize Function

Generates generalized template from a set of templates.

```
NResult N\_API FcceGeneralize(
    HFccExtractor hExtractor,
    NInt templateCount,
    const void **arTemplates,
    const NSizeType *arTemplateSizes,
    void *pBuffer,
    NSizeType bufferSize,
    NSizeType *pSize
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
<i>templateCount</i>	[in] Count of templates to be used in generalization.
<i>arTemplates</i>	[in] pointer to array of the templates to be used in generalization.
<i>pBuffer</i>	[out] address to the memory space allocated for the generalized features.
<i>bufferSize</i>	[in] Size of the allocated buffer for the extracted features.
<i>pSize</i>	[out] Returns number of bytes filled in the buffer.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> or <i>arTemplates</i> or <i>arTemplateSizes</i> or <i>pBuffer</i> or <i>pSize</i> is NULL .
N_E_ARGUMENT	One or more of templates in <i>arTemplates</i> might be NULL .

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.15. FcceGetMaxTemplateSize Function

Retrieves maximal size of feature template that the specified FccExtractor can extract.

```
NResult N_API FcceGetMaxTemplateSize(
    HFccExtractor hExtractor,
    NSizeType * pSize
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
<i>pSize</i>	[out] Pointer to NSizeType that receives maximal template size.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> or <i>pSize</i> is NULL .

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.16. FcceGetParameter Function

Retrieves value of the specified parameter of the specified FccExtractor.

```
NResult N_API FcceGetParameter(
    HFccExtractor hExtractor,
    NUInt parameterId,
    void * pValue
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object. Can be NULL if retrieving static parameter value.
<i>parameterId</i>	[in] Identifier of the parameter to retrieve.
<i>pValue</i>	[out] Pointer to variable that receives parameter value.

Return Values

If the function succeeds and *parameterId* specifies a [N_TYPE_STRING](#) type parameter, and *pValue* is [NULL](#), the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not [NULL](#), the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>parameterId</i> specifies a non-static parameter and <i>hExtractor</i> is NULL . - or - <i>parameterId</i> specifies a non- N_TYPE_STRING type parameter and <i>pValue</i> is NULL .
N_E_PARAMETER	<i>parameterId</i> is invalid.

Remarks

The following values can be used for *parameterId*:

- [FCCEP_MIN_IOD](#)
- [FCCEP_MAX_IOD](#)
- [FCCEP_FACE_CONFIDENCE_THRESHOLD](#)
- [FCCEP_GENERALIZATION_THRESHOLD](#)
- [FCCEP_NAME](#)
- [FCCEP_VERSION_HIGH](#)
- [FCCEP_VERSION_LOW](#)
- [FCCEP_COPYRIGHT](#)

To learn the type of the parameter pass value obtained with [NParameterMakeId](#) macro using [N_PC_TYPE_ID](#) code and the parameter id via *parameterId* parameter and pointer to [NInt](#) that will receive one of [N_TYPE_XXX](#) via *pValue* parameter. *hExtractor* can be [NULL](#) in this case.

See Also

[FccExtractor Module](#) | [HFccExtractor](#) | [FcceSetParameter](#)

6.1.1.17. FccIsRegistered Function

Checks if FccExtractor library is registered.

```
NBool N_API FcceIsRegistered(void);
```

Return Values

[NTrue](#) if library is registered, [NFalse](#) otherwise.

See Also

[FccExtractor Module](#)

6.1.1.18. FcceReset Function

Sets default values for all parameters of the specified FccExtractor.

```
NResult N_API FcceReset(
    HFccExtractor hExtractor
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object.
-------------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hExtractor</i> is NULL .

See Also

[FccExtractor Module](#) | [HFccExtractor](#)

6.1.1.19. FcceSetParameter Function

Sets value of the specified parameter of the specified FccExtractor.

```
NResult N_API FcceSetParameter(
    HFccExtractor hExtractor,
    NUInt parameterId,
    const void * pValue
);
```

Parameters

<i>hExtractor</i>	[in] Handle to the FccExtractor object. Can be NULL if setting static parameter value.
<i>parameterId</i>	[in] Identifier of the parameter to set.
<i>pValue</i>	[in] Pointer to the parameter value to set.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	Argument value is out of range.
N_E_PARAMETER	Parameter ID is invalid.

Remarks

The following values can be used for *parameterId*:

- [FCCEP_MIN_IOD](#)
- [FCCEP_MAX_IOD](#)
- [FCCEP_FACE_CONFIDENCE_THRESHOLD](#)
- [FCCEP_GENERALIZATION_THRESHOLD](#)

To learn the type of the parameter pass value obtained with [NParameterMakeId](#) macro using [N_PC_TYPE_ID](#) code and the parameter id via *parameterId* parameter and pointer to [NInt](#) that will receive one of [N_TYPE_XXX](#) via *pValue* parameter, *hExtractor* can be [NULL](#) in this case.

See Also

[FccExtractor Module](#) | [HFccExtractor](#) | [FcceGetParameter](#)

6.2. FccMatcher Library

Provides functionality to verify and match VeriLook face feature templates.

Windows

Import library: `FccMatcher.dll.lib`.

DLL: `FccMatcher.dll`.

Requirements:

- [NCore.dll](#).

Linux

Shared object: `libFccMatcher.so`.

Requirements:

- [libNCore.so](#).

Modules

FccMatcher	Provides functionality to identify or verify VeriLook face feature templates.
----------------------------	---

6.2.1. FccMatcher Module

Provides functionality to identify or verify VeriLook face feature templates.

Header file: `FccMatcher.h` (includes `FccMatcherParams.h` and `FccMatcherTypes.h`)

Functions

FccmCopyParameters	Copies parameter values from one FccMatcher to another.
------------------------------------	---

FccmCreate	Creates a FccMatcher.
FccmFree	Deletes the FccMatcher. After the object is deleted the specified handle is no longer valid.
FccmGetParameter	Retrieves value of the specified parameter of the specified FccMatcher.
FccmIdentifyEnd	Finalizes identification procedure.
FccmIdentifyNext	Verifies one given template with the one that is being identified.
FccmIdentifyStart	Initializes identification procedure.
FccmIsRegistered	Checks if FccMatcher library is registered.
FccmReset	Sets default values for all parameters of the specified FccMatcher.
FccmSetParameter	Sets value of the specified parameter of the specified FccMatcher.
FccmVerify	Verifies two VeriLook face feature templates and return their similarity score.

Types

HFccMatcher	Handle to FccMatcher object.
-----------------------------	------------------------------

Macros

FCCMP_COPYRIGHT	Identifier specifying library copyright static read-only parameter of type N_TYPE_STRING .
FCCMP_MATCHING_THRESHOLD	Template matching threshold. For possible values of this parameter see Thresholds and similarities section.
FCCMP_NAME	Identifier specifying library name static read-only parameter of type N_TYPE_STRING .
FCCMP_VERSION_HIGH	Identifier specifying high part of library version static read-only parameter of type N_TYPE_UINT . Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
FCCMP_VERSION_LOW	Identifier specifying low part of library version static read-only parameter of type N_TYPE_UINT . Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

See Also

[FccMatcher Library](#)

6.2.1.1. FccmCopyParameters Function

Copies parameter values from one FccMatcher to another.

```
NResult N_API FccmCopyParameters(
    HFccMatcher hDstMatcher,
```

```

    HFccMatcher hSrcMatcher
);

```

Parameters

<i>hDstMatcher</i>	[in] Handle to the destination FccMatcher object.
<i>hSrcMatcher</i>	[in] Handle to the source FccMatcher object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hDstMatcher</i> or <i>hSrcMatcher</i> is NULL .

See Also

[FccMatcher Module](#) | [HFccMatcher](#)

6.2.1.2. FccmCreate Function

Creates a FccMatcher.

```

NResult N_API FccmCreate(
    HFccMatcher * pHMatcher
);

```

Parameters

<i>pHMatcher</i>	[out] Pointer to HFccMatcher that receives handle to created FccMatcher object.
------------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHMatcher</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [FccmFree](#) function.

See Also

[FccMatcher Module](#) | [HFccMatcher](#) | [FccmFree](#)

6.2.1.3. FccmFree Function

Deletes the FccMatcher. After the object is deleted the specified handle is no longer valid.

```
void N_API FccmFree(
    HFccMatcher hMatcher
);
```

Parameters

<i>hMatcher</i>	[in] Handle to FccMatcher object.
-----------------	-----------------------------------

Remarks

If *hMatcher* is [NULL](#), does nothing.

See Also

[FccMatcher Module](#) | [HFccMatcher](#)

6.2.1.4. FccmGetParameter Function

Retrieves value of the specified parameter of the specified FccMatcher.

```
NResult N_API FccmGetParameter(
    HFccMatcher hMatcher,
    NUInt parameterId,
    void * pValue
);
```

Parameters

<i>hMatcher</i>	[in] Handle to the FccMatcher object. Can be NULL if retrieving static parameter value.
<i>parameterId</i>	[in] Identifier of the parameter to retrieve.
<i>pValue</i>	[out] Pointer to variable that receives parameter value.

Return Values

If the function succeeds and *parameterId* specifies a [N_TYPE_STRING](#) type parameter, and *pValue* is [NULL](#), the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not [NULL](#), the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>parameterId</i> specifies a non-static parameter and <i>hMatcher</i> is NULL . - or - <i>parameterId</i> specifies a non- N_TYPE_STRING type parameter and <i>pValue</i> is NULL .
N_E_PARAMETER	<i>parameterId</i> is invalid.

Remarks

The following values can be used for *parameterId*:

- [FCCMP_MATCHING_THRESHOLD](#)
- [FCCMP_NAME](#)
- [FCCMP_COPYRIGHT](#)
- [FCCMP_VERSION_HIGH](#)
- [FCCMP_VERSION_LOW](#)

To learn the type of the parameter pass value obtained with [NParameterMakeId](#) macro using [N_PC_TYPE_ID](#) code and the parameter id via *parameterId* parameter and pointer to [NInt](#) that will receive one of [N_TYPE_XXX](#) via *pValue* parameter. *hMatcher* can be [NULL](#) in this case.

See Also

[FccMatcher Module](#) | [HFccMatcher](#) | [FccmSetParameter](#)

6.2.1.5. FccmIdentifyStart Function

Initializes identification procedure.

```
NResult N_API FccmIdentifyStart(
    HFccMatcher hMatcher,
    const void *pTemplate,
    NSizeType templateSize,
);
```

Parameters

<i>hMatcher</i>	[in] Handle to the FccMatcher object.
<i>pTemplate</i>	[in] Pointer to memory buffer containing face features template.
<i>templateSize</i>	[in] Size of given template.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hMatcher</i> or <i>pTemplate</i> is NULL or wrong size.
N_E_INVALID_OPERATION	Identification is already started.

Remarks

FccmIdentifyStart takes the template to be identified, which is then matched against the templates from the user database using [FccmIdentifyNext](#).

See Also

[FccMatcher Module](#) | [HFccMatcher](#) | [FccmIdentifyNext](#) | [FccmIdentifyEnd](#)

6.2.1.6. FccmIdentifyNext Function

Verifies one given template with the one that is being identified.

```
NResult N_API FccmIdentifyNext(
```

```

    HFccMatcher hMatcher,
    const void * pTemplate,
    NSizeType templateSize,
    NDouble * pScore
);

```

Parameters

<i>hMatcher</i>	[in] Handle to the FccMatcher object.
<i>pTemplate</i>	[in] Pointer to memory buffer containing face features template.
<i>templateSize</i>	[in] Size of given template.
<i>pScore</i>	[out] Pointer to NDouble that receives similarity score.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hMatcher</i> , <i>pTemplate</i> , or <i>pScore</i> is NULL or <i>pTemplate</i> is wrong size.
N_E_INVALID_OPERATION	Identification is not started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two templates do not match (see [FCCMP_MATCHING_THRESHOLD](#) and [FccmSetParameter](#) function), and is greater than or equal to matching threshold otherwise.

See Also

[FccMatcher Module](#) | [HFccMatcher](#) | [FCCMP_MATCHING_THRESHOLD](#) | [FccmSetParameter](#) | [FccmIdentifyStart](#) | [FccmIdentifyEnd](#)

6.2.1.7. FccmIdentifyEnd Function

Finalizes identification procedure.

```

NResult N_API FccmIdentifyEnd(
    HFccMatcher hMatcher
);

```

Parameters

<i>hMatcher</i>	[in] Handle to the FccMatcher object.
-----------------	---------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hMatcher</i> is NULL .
N_E_INVALID_OPERATION	Identification is not started.

See Also

[FccMatcher Module](#) | [HFccMatcher](#) | [FccmIdentifyStart](#) | [FccmIdentifyNext](#)

6.2.1.8. FccmIsRegistered Function

Checks if FccMatcher library is registered.

```
NBool N_API FccmIsRegistered(void);
```

Return Values

[NTrue](#) if library is registered, [NFalse](#) otherwise.

See Also

[FccMatcher Module](#)

6.2.1.9. FccmReset Function

Sets default values for all parameters of the specified FccMatcher.

```
NResult N_API FccmReset(
    HFccMatcher hMatcher
);
```

Parameters

<i>hMatcher</i>	[in] Handle to VIMatcher object.
-----------------	----------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hMatcher</i> is NULL .

See Also

[FccMatcher Module](#) | [HFccMatcher](#)

6.2.1.10. FccmSetParameter Function

Sets value of the specified parameter of the specified FccMatcher.

```
NResult N_API FccmSetParameter(
    HFccMatcher hMatcher,
    NUInt parameterId,
    const void * pValue
```

```
);
```

Parameters

<i>hMatcher</i>	[in] Handle to the FccMatcher object. Can be NULL if setting static parameter value.
<i>parameterId</i>	[in] Identifier of the parameter to set.
<i>pValue</i>	[in] Pointer to the parameter value to set.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>parameterId</i> specifies a non-static parameter and <i>hMatcher</i> is NULL . - or - <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	Value <i>pValue</i> points to is out of range.
N_E_INVALID_OPERATION	<i>hMatcher</i> is not NULL and identification is started on the matcher.
N_E_PARAMETER	<i>parameterId</i> is invalid.
N_E_PARAMETER_READ_ONLY	<i>parameterId</i> specifies read-only parameter.

Remarks

The following values can be used for *parameterId*:

- [FCCMP_MATCHING_THRESHOLD](#)

To learn the type of the parameter pass value obtained with [NParameterMakeId](#) macro using [N_PC_TYPE_ID](#) code and the parameter id via *parameterId* parameter and pointer to [NInt](#) that will receive one of [N_TYPE_XXX](#) via *pValue* parameter, *hMatcher* can be [NULL](#) in this case.

See Also

[FccMatcher Module](#) | [HFccMatcher](#) | [FccmGetParameter](#)

6.2.1.11. FccmVerify Function

Verifies two VeriLook face feature templates and return their similarity score.

```
NResult N_API FccmVerify(
    HFccMatcher hMatcher,
    const void * pTemplate1,
    NSizeType template1Size,
    const void * pTemplate2,
    NSizeType template2Size,
    NDouble * pScore
```



```
);
```

Parameters

<i>hMatcher</i>	[in] Handle to the FccMatcher object.
<i>template1</i>	[in] Pointer to memory buffer containing first face features template.
<i>template1Size</i>	[in] Size of first face features template.
<i>template2</i>	[in] Pointer to memory buffer containing second face features template.
<i>template2Size</i>	[in] Size of second face features template.
<i>pScore</i>	[out] Pointer to NDouble that receives similarity score.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hMatcher</i> or <i>pTemplate1</i> or <i>pTemplate2</i> is NULL or <i>pTemplate1</i> or <i>pTemplate2</i> is wrong size.
N_E_INVALID_OPERATION	Identification is already started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two face templates do not match (see [FCCMP_MATCHING_THRESHOLD](#) and [FccmSetParameter](#) function), and is greater than or equal to matching threshold otherwise.

See Also

[FccMatcher Module](#) | [HFccMatcher](#) | [FCCMP_MATCHING_THRESHOLD](#) | [FccmSetParameter](#) | [FccmIdentifyStart](#) | [FccmIdentifyNext](#) | [FccmIdentifyEnd](#)

6.3. NCore Library

Provides infrastructure for Neurotechnology components. [NErrors](#) defines error codes used in Neurotechnology components.

Import library: `NCore.dll.lib`.

DLL: `NCore.dll`.

Linux

Shared object: `libNCore.so`.

Modules

NCore	Provides infrastructure/basic functionality for Neurotechnology components.
NErrors	Defines error codes used in Neurotechnology components.

NGeometry	Provides definitions of geometrical structures types.
NLibraryInfo	Provides definitions of library info structure type.
NMemory	Provides memory management for Neurotechnology components.
NParameters	Provides functionality for working with parameters for Neurotechnology components.
NTypes	Defines types and macros used in Neurotechnology components.

6.3.1. NCore Module

Provides infrastructure/basic functionality for Neurotechnology components.

Header file: `NCore.h`.

Functions

NCoreGetInfo	Gets information about the library.
------------------------------	-------------------------------------

See Also

[NCore Library](#)

6.3.1.1. NCoreGetInfo Function

Gets information about the library.

```
NResult N_API NCoreGetInfo(
    NLibraryInfo * pValue
);
```

Parameters

<i>pValue</i>	[out] Pointer to NLibraryInfo structure that receives library information.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	pValue is NULL .

6.3.2. NErrors Module

Defines error codes used in Neurotechnology components.

Header file: `NErrors.h`.

Macros

0	N_OK	No error.
-1	N_E_FAILED	Unspecified error has occurred.
-2	N_E_CORE	Standard error has occurred (for internal use).
-3	N_E_NULL_REFERENCE	Null reference has occurred (for internal use).
-4	N_E_OUT_OF_MEMORY	There were not enough memory.
-5	N_E_NOT_IMPLEMENTED	Functionality is not implemented.
-6	N_E_NOT_SUPPORTED	Functionality is not supported.
-7	N_E_INVALID_OPERATION	Attempted to perform invalid operation.
-8	N_E_OVERFLOW	Arithmetic overflow has occurred.
-9	N_E_INDEX_OUT_OF_RANGE	Index is out of range (for internal use).
-10	N_E_ARGUMENT	Argument is invalid.
-11	N_E_ARGUMENT_NULL	Argument value is NULL where non-NULL value was expected.
-12	N_E_ARGUMENT_OUT_OF_RANGE	Argument value is out of range.
-13	N_E_FORMAT	Format of argument value is invalid.
-14	N_E_IO	Input/output error has occurred.
-15	N_E_END_OF_STREAM	Attempted to read file or buffer after its end.
-90	N_E_EXTERNAL	Error in external code has occurred (for internal use).
-91	N_E_WIN32	Win32 error has occurred.
-92	N_E_COM	COM error has occurred.
-93	N_E_CLR	CLR exception has occurred.
-100	N_E_PARAMETER	Parameter ID is invalid.
-101	N_E_PARAMETER_READ_ONLY	Attempted to set read only parameter.
-200	N_E_NOT_REGISTERED	Module is not registered.
	NFailed	Determines whether function result indicates error.
	NSucceeded	Determines whether function result indicates success.

See Also

[NCore Library](#)

6.3.3. NGeometry Module

Provides definitions of geometrical structures types.

Header file: `NGeometry.h`.

Structures

NPoint	Structure defining point coordinates in 2D space.
NSize	Structure defining rectangle size.
NRect	Structure defining a rectangle figure in 2D space.

See Also

[NCore Library](#)

6.3.3.1. NPoint structure

Structure defining point coordinates in 2D space.

```
typedef struct NPoint_ { } NPoint;
```

Fields

X	Point coordinate on x axis.
Y	Point coordinate on y axis.

See Also

[NGeometry](#)

6.3.3.1.1. NPoint.X Field

Point coordinate on x axis.

```
NInt X;
```

See Also

[NPoint](#)

6.3.3.1.2. NPoint.Y Field

Point coordinate on y axis.

```
NInt Y;
```

See Also

[NPoint](#)

6.3.3.2. NSize structure

Structure defining rectangle size.

```
typedef struct NSize_ { } NSize;
```

Fields

Width	Width.
Height	Height.

See Also[NGeometry](#)**6.3.3.2.1. NSize.Width Field**

Width.

```
NInt Width;
```

See Also[NSize](#)**6.3.3.2.2. NSize.Height Field**

Height.

```
NInt Height;
```

See Also[NSize](#)**6.3.3.3. NRect structure**

Structure defining a rectangle figure in 2D space.

```
typedef struct NRect_ { } NRect;
```

Fields

<i>X</i>	Upper left rectangle corner coordinate on x axis.
<i>Y</i>	Upper left rectangle corner coordinate on y axis.
<i>Width</i>	Rectangle width.
<i>Height</i>	Rectangle height.

See Also[NGeometry](#)**6.3.3.3.1. NRect.X Field**

Upper left rectangle corner coordinate on x axis.

```
NInt X;
```

See Also[NRect](#)**6.3.3.3.2. NRect.Y Field**

Upper left rectangle corner coordinate on y axis.

```
NInt Y;
```

See Also[NRect](#)**6.3.3.3. NRect.Width Field**

Rectangle width.

```
NInt Width;
```

See Also[NRect](#)**6.3.3.3.4. NRect.Height Field**

Rectangle height.

```
NInt Height;
```

See Also[NRect](#)**6.3.4. NLibraryInfo Module**

Provides definitions of library info structure type.

Header file: `NLibraryInfo.h`.

Structures

NLibraryInfo	Structure defining information about the library as library title, product name, company name, copyright string and library version.
------------------------------	--

See Also[NCore Library](#)**6.3.4.1. NLibraryInfo structure**

Structure defining information about the library as library title, product name, company name, copyright string and library version.

```
typedef struct NLibraryInfo_ { } NLibraryInfo;
```

Fields

Company	Name of the company that produced the library.
Copyright	Copyright string for the library.
Product	Product name.
Title	Title of the library.
VersionBuild	Build part of the library version.
VersionMajor	Major part of the library version.

VersionMinor	Minor part of the library version.
VersionRevision	Revision part of the library version.

See Also

[NLibraryInfo](#)

6.3.4.1.1. NLibraryInfo.Company Field

Name of the company that produced the library.

```
NChar * Company;
```

See Also

[NLibraryInfo](#)

6.3.4.1.2. NLibraryInfo.Copyright Field

Copyright string for the library.

```
NChar * Copyright;
```

See Also

[NLibraryInfo](#)

6.3.4.1.3. NLibraryInfo.Product Field

Product name.

```
NChar * Product;
```

See Also

[NLibraryInfo](#)

6.3.4.1.4. NLibraryInfo.Title Field

Title of the library.

```
NChar * Title;
```

See Also

[NLibraryInfo](#)

6.3.4.1.5. NLibraryInfo.VersionBuild Field

Build part of the library version.

```
NInt VersionBuild;
```

See Also

[NLibraryInfo](#)

6.3.4.1.6. NLibraryInfo.VersionMajor Field

Major part of the library version.

```
NInt VersionMajor;
```

See Also

[NLibraryInfo](#)

6.3.4.1.7. NLibraryInfo.VersionMinor Field

Minor part of the library version.

```
NInt VersionMinor;
```

See Also

[NLibraryInfo](#)

6.3.4.1.8. NLibraryInfo.VersionRevision Field

Revision part of the library version.

```
NInt VersionRevision;
```

See Also

[NLibraryInfo](#)

6.3.5. NMemory Module

Provides memory management for Neurotechnology components.

Header file: `NMemory.h`.

Functions

<code>NAlloc</code>	Allocates memory block.
<code>NAlloc</code>	Allocates memory block with all bytes set to zero.
<code>NCompare</code>	Compares bytes in two memory blocks.
<code>NCopy</code>	Copies data between memory blocks.
<code>NFill</code>	Sets bytes of memory block to specified value.
<code>NFree</code>	Deallocates memory block.
<code>NMove</code>	Move data from one memory block to another.
<code>NReAlloc</code>	Reallocates memory block.

Macros

<code>NClear</code>	Clears memory block.
---------------------	----------------------

See Also

[NCore Library](#)

6.3.6. NParameters Module

Provides functionality for working with parameters for Neurotechnology components.

Header file: `NParameters.h`.

Macros

<code>N_PC_TYPE_ID</code>	Specifies that type id (NInt value, one of <code>N_TYPE_XXX</code>) of the parameter should be retrieved.
NParameterMakeId	Makes parameter id.
<code>N_TYPE_BOOL</code>	Specifies that parameter type is NBool .
<code>N_TYPE_BYTE</code>	Specifies that parameter type is NByte .
<code>N_TYPE_CHAR</code>	Specifies that parameter type is NChar .
<code>N_TYPE_DOUBLE</code>	Specifies that parameter type is NDouble .
<code>N_TYPE_FLOAT</code>	Specifies that parameter type is NFloat .
<code>N_TYPE_INT</code>	Specifies that parameter type is NInt .
<code>N_TYPE_LONG</code>	Specifies that parameter type is NLong .
<code>N_TYPE_SBYTE</code>	Specifies that parameter type is NSByte .
<code>N_TYPE_SHORT</code>	Specifies that parameter type is NShort .
<code>N_TYPE_STRING</code>	Specifies that parameter type is null-terminated string of NChar .
<code>N_TYPE_UINT</code>	Specifies that parameter type is NUInt .
<code>N_TYPE_ULONG</code>	Specifies that parameter type is NULong .
<code>N_TYPE_USHORT</code>	Specifies that parameter type is NUShort .

See Also

[NCore Library](#)

6.3.6.1. NParameterMakeId Macro

Makes parameter id.

```
#define NParameterMakeId(code, index, id)
```

Parameters

<i>code</i>	One of <code>N_PC_XXX</code> .
<i>index</i>	Reserved, must be zero.
<i>id</i>	One of the parameter ids provided by a Neurotechnology module.

See Also

[NParameters Module](#)

6.3.7. NTypes Module

Defines types and macros used in Neurotechnology components.

Header file: `NTypes.h`.

Structures

NIndexPair	Represents a pair of indexes.
NRational	Represents a signed rational number.
NURational	Represents an unsigned rational number.

Enumerations

NByteOrder	Specifies byte order.
NFileAccess	Specifies access to a file.

Types

<code>NChar</code>	ANSI character (8-bit).
<code>NBool</code>	Same as NBoolean .
<code>NBoolean</code>	32-bit boolean value. See also NTrue and NFalse .
<code>NByte</code>	Same as NUInt8 .
<code>NChar</code>	Character type. Either NChar or NWChar (if N_UNICODE is defined).
<code>NDouble</code>	Double precision floating point number.
<code>NFloat</code>	Same as NSingle .
<code>NHandle</code>	Pointer to unspecified data (same as <code>void *</code>).
<code>NInt</code>	Same as NInt32 .
<code>NInt8</code>	8-bit signed integer (signed byte).
<code>NInt16</code>	16-bit signed integer (short).
<code>NInt32</code>	32-bit signed integer (int).
<code>NInt64</code>	64-bit signed integer (long). Not available on some 32-bit platforms.
<code>NLong</code>	Same as NInt64 .
<code>NPosType</code>	Platform dependent position type. Signed 64-bit (or 32-bit on some platforms) integer on 32-bit platform, signed 64-bit integer on 64-bit platform).
<code>NResult</code>	Result of a function (same as NInt). See also NErrors module.
<code>NSByte</code>	Same as NInt8 .
<code>NShort</code>	Same as NInt16 .
<code>NSingle</code>	Single precision floating point number.

NSizeType	Platform dependent size type. Unsigned 32-bit integer on 32-bit platform, unsigned 64-bit integer on 64-bit platform.
NUInt	Same as NUInt32 .
NUInt8	8-bit unsigned integer (byte).
NUInt16	16-bit unsigned integer (unsigned short).
NUInt32	32-bit unsigned integer (unsigned int).
NUInt64	64-bit unsigned integer (unsigned long). Not available on some 32-bit platforms.
NULong	Same as NUInt64 .
NUShort	Same as NUInt16 .
NWChar	Unicode character (16-bit).

Macros

N_64	Defined if compiling for 64-bit architecture.
N_ANSI_C	Defined if ANSI C language compliance is enabled in compiler.
N_API	Defines functions calling convention (stdcall on Windows).
N_BIG_ENDIAN	Defined if compiling for big-endian processor architecture.
N_BYTE_MAX	Maximum value for NByte .
N_BYTE_MIN	Minimum value for NByte .
N_CALLBACK	Defined callbacks calling convention (stdcall on Windows).
N_CALLBACK_AW	Picks either ANSI or Unicode (if N_UNICODE is defined) version of the callback (with either 'A' or 'W' suffix accordingly).
N_CPP	Defined if compiling as C++ code.
N_DECLARE_HANDLE	Declares handle with specified name.
N_DOUBLE_MAX	Maximum value for NDouble .
N_DOUBLE_MIN	Minimum value for NDouble .
N_FUNC_AW	Picks either ANSI or Unicode (if N_UNICODE is defined) version of the function (with either 'A' or 'W' suffix accordingly).
N_GCC	Defined if compiling with GCC.
N_FLOAT_MAX	Maximum value for NFloat .
N_FLOAT_MIN	Minimum value for NFloat .
N_INT_MAX	Maximum value for NInt .
N_INT_MIN	Minimum value for NInt .

N_INT8_MAX	Maximum value for NInt8 .
N_INT8_MIN	Minimum value for NInt8 .
N_INT16_MAX	Maximum value for NInt16 .
N_INT16_MIN	Minimum value for NInt16 .
N_INT32_MAX	Maximum value for NInt32 .
N_INT32_MIN	Minimum value for NInt32 .
N_INT64_MAX	Maximum value for NInt64 .
N_INT64_MIN	Minimum value for NInt64 .
N_LIB	Defined if compiling static library.
N_LONG_MAX	Maximum value for NLong .
N_LONG_MIN	Minimum value for NLong .
N_MAC	Defined if compiling for Mac OS.
N_MSVC	Defined if compiling with Microsoft Visual C++.
N_NO_ANSI_FUNC	Defined if compiling for platform without ANSI versions of the functions support.
N_NO_FLOAT	Defined if compiling for platform without floating-point support.
N_NO_INT_64	Defined if compiling for platform without 64-bit integer types support.
N_NO_UNICODE	Defined if compiling without Unicode support.
N_POS_TYPE_MIN	Minimum value for NPosType .
N_POS_TYPE_MAX	Maximum value for NPosType .
N_SBYTE_MAX	Maximum value for NSByte .
N_SBYTE_MIN	Minimum value for NSByte .
N_SHORT_MAX	Maximum value for NShort .
N_SHORT_MIN	Minimum value for NShort .
N_SINGLE_MAX	Maximum value for NSingle .
N_SINGLE_MIN	Minimum value for NSingle .
N_SIZE_TYPE_MIN	Minimum value for NSizeType .
N_SIZE_TYPE_MAX	Maximum value for NSizeType .
N_STRUCT_AW	Picks either ANSI or Unicode (if N_UNICODE is defined) version of the struct (with either 'A' or 'W' suffix accordingly).
N_T	Makes either ANSI or Unicode (if N_UNICODE is defined) string or character constant.
N_UINT_MAX	Maximum value for NUInt .
N_UINT_MIN	Minimum value for NUInt .

N_UINT8_MAX	Maximum value for NUInt8 .
N_UINT8_MIN	Minimum value for NUInt8 .
N_UINT16_MAX	Maximum value for NUInt16 .
N_UINT16_MIN	Minimum value for NUInt16 .
N_UINT32_MAX	Maximum value for NUInt32 .
N_UINT32_MIN	Minimum value for NUInt32 .
N_UINT64_MAX	Maximum value for NUInt64 .
N_UINT64_MIN	Minimum value for NUInt64 .
N_ULONG_MAX	Maximum value for NULong .
N_ULONG_MIN	Minimum value for NULong .
N_UNICODE	Defined if compiling with Unicode character set (affects NChar type).
N_USHORT_MAX	Maximum value for NUShort .
N_USHORT_MIN	Minimum value for NUShort .
N_WINDOWS	Defined if compiling for Windows.
NULL	Null value for pointer.
NFalse	False value for NBoolean .
NIsReverseByteOrder	Checks if specified byte order is reverse to system byte order.
NTrue	True value for NBoolean .

See Also

[NCore Library](#)

6.3.7.1. NByteOrder Enumeration

Specifies byte order.

```
typedef enum NByteOrder_ { } NByteOrder;
```

Members

nboBigEndian	Big-endian byte order.
nboLittleEndian	Little-endian byte order.
nboSystem	System-dependent byte order (either little-endian or big-endian).

See Also

[NTypes Module](#)

6.3.7.2. NFileAccess Enumeration

Specifies access to a file.

```
typedef enum NFileAccess_ { } NFileAccess;
```

Members

nfaRead	Read access to the file.
nfaReadWrite	Read and write access to the file.
nfaWrite	Write access to the file.

See Also

[NTypes Module](#)

6.3.7.3. NIndexPair Structure

Represents a pair of indexes.

```
typedef struct NIndexPair_ { } NIndexPair;
```

Fields

Index1	First index of this NIndexPair .
Index2	Second index of this NIndexPair .

See Also

[NTypes Module](#)

6.3.7.3.1. NIndexPair.Index1 Field

First index of this [NIndexPair](#).

```
NInt Index1;
```

See Also

[NIndexPair](#)

6.3.7.3.2. NIndexPair.Index2 Field

Second index of this [NIndexPair](#).

```
NInt Index2;
```

See Also

[NIndexPair](#)

6.3.7.4. NRational Structure

Represents a signed rational number.

```
typedef struct NRational_ { } NRational;
```

Fields

<i>Denominator</i>	Denominator of this NRational .
<i>Numerator</i>	Numerator of this NRational .

See Also[NTypes Module](#)**6.3.7.4.1. NRational.Denominator Field**Denominator of this [NRational](#).

```
NInt Denominator;
```

See Also[NRational](#)**6.3.7.4.2. NRational.Numerator Field**Numerator of this [NRational](#).

```
NInt Numerator;
```

See Also[NRational](#)**6.3.7.5. NURational Structure**

Represents an unsigned rational number.

```
typedef struct NURational_ { } NURational;
```

Fields

<i>Denominator</i>	Denominator of this NURational .
<i>Numerator</i>	Numerator of this NURational .

See Also[NTypes Module](#)**6.3.7.5.1. NURational.Denominator Field**Denominator of this [NURational](#).

```
NUInt Denominator;
```

See Also[NURational](#)**6.3.7.5.2. NURational.Numerator Field**Numerator of this [NURational](#).

```
NUInt Numerator;
```

See Also

[NURational](#)

6.4. NImages Library

Provides functionality for loading, saving and converting images in various formats.

Import library: `NImages.dll.lib`.

DLL: `NImages.dll`.

Requirements:

- [NCore.dll](#).

Linux

Shared object: `libNImages.so`.

Requirements:

- [libNCore.so](#).

Modules

Bmp	Provides functionality for loading and saving images in BMP format.
Jpeg	Provides functionality for loading and saving images in JPEG format.
NGrayscaleImage	Provides functionality for managing 8-bit grayscale images.
NImageFormat	Provides functionality for loading and saving images in format-neutral way.
NImage	Provides functionality for managing images.
NImages	Provides library registration and other additional functionality.
NMonochromeImage	Provides functionality for managing 1-bit monochrome images.
NPixelFormat	Provides functionality for working with image pixel format.
NRGBImage	Provides functionality for managing 24-bit RGB images.
Png	Provides functionality for loading and saving images in PNG format.
Tiff	Provides functionality for loading images in TIFF format.

6.4.1. Bmp Module

Provides functionality for loading and saving images in BMP format.

Header file: `Bmp.h`.

Functions

BmpLoadImageFromFile	Loads image from BMP file.
BmpLoadImageFromHBitmap	Loads image from Windows HBITMAP.
BmpLoadImageFromMemory	Loads image from memory buffer containing BMP file.
BmpSaveImageToFile	Saves image to file in BMP format.
BmpSaveImageToHBitmap	Saves image to Windows HBITMAP.
BmpSaveImageToMemory	Saves image to memory buffer in BMP format.

See Also

[NImages Library](#)

6.4.1.1. BmpLoadImageFromFile Function

Loads image from BMP file.

```
NResult N_API BmpLoadImageFromFile(
    const NChar * szFileName,
    HImage * pImage
);
```

Parameters

<i>szFileName</i>	[in] Points to string that specifies file name.
<i>pImage</i>	[out] Pointer to HImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>szFileName</i> or <i>pImage</i> is NULL .
N_E_FORMAT	Format of file specified by <i>szFileName</i> is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

[Bmp Module](#) | [HImage](#) | [BmpLoadImageFromMemory](#) | [BmpLoadImageFromHBitmap](#) | [BmpSaveImageToFile](#)

6.4.1.2. BmpLoadImageFromHBitmap Function

Note

This function is available only on Windows.

Loads image from Windows HBITMAP.

```
NResult N_API BmpLoadImageFromHBitmap(
    NHandle handle,
    HNImage * pImage
);
```

Parameters

<i>handle</i>	[in] Handle that specifies Windows HBITMAP.
<i>pImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>handle</i> or <i>pImage</i> is NULL .

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

[Bmp Module](#) | [HNImage](#) | [BmpLoadImageFromFile](#) | [BmpLoadImageFromMemory](#) | [BmpSaveImageToHBitmap](#)

6.4.1.3. BmpLoadImageFromMemory Function

Loads image from memory buffer containing BMP file.

```
NResult N_API BmpLoadImageFromMemory(
    const void * buffer,
    NSizeType bufferLength,
    HNImage * pImage
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer.
<i>bufferLength</i>	[in] Length of memory buffer.
<i>pImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL and <i>bufferLength</i> is not equal to zero. - or - <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file contained in buffer specified by <i>buffer</i> is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

[Bmp Module](#) | [HNImage](#) | [BmpLoadImageFromFile](#) | [BmpLoadImageFromHBitmap](#) | [BmpSaveImageToMemory](#)

6.4.1.4. BmpSaveImageToFile Function

Saves image to file in BMP format.

```
NResult N_API BmpSaveImageToFile(
    HNImage hImage,
    const NChar * szFileName
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>szFileName</i>	[in] Points to string that specifies file name.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>szFileName</i> is NULL .

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

[Bmp Module](#) | [HNImage](#) | [BmpSaveImageToMemory](#) | [BmpSaveImageToHBitmap](#) | [BmpLoadImageFromFile](#)

6.4.1.5. BmpSaveImageToHBitmap Function

Note

This function is available only on Windows.

Saves image to Windows HBITMAP.

```
NResult N_API BmpSaveImageToHBitmap(
    HImage hImage,
    NHandle * pHandle
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>pHandle</i>	[out] Pointer to NHandle that receives handle to created Windows HBITMAP.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pHandle</i> is NULL .

Remarks

This is a low-level function and can be changed in future version of the library.

HBITMAP must be freed using GDI function DeleteObject.

See Also

[Bmp Module](#) | [HImage](#) | [BmpSaveImageToFile](#) | [BmpSaveImageToMemory](#) | [BmpLoadImageFromHBitmap](#)

6.4.1.6. BmpSaveImageToMemory Function

Saves image to memory buffer in BMP format.

```
NResult N_API BmpSaveImageToMemory(
    HImage hImage,
    void * * pBuffer,
    NSizeType * pBufferLength
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>pBuffer</i>	[out] Pointer to void * that receives pointer to allocated memory buffer.
<i>pBufferLength</i>	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> , <i>pBuffer</i> or <i>pBufferLength</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory to allocate memory buffer.

Remarks

This is a low-level function and can be changed in future version of the library.

Memory buffer allocated by the function must be deallocated using [NFree](#) function when it is no longer needed.

See Also

[Bmp Module](#) | [HNImage](#) | [BmpSaveImageToFile](#) | [BmpSaveImageToHBitmap](#) | [BmpLoadImageFromMemory](#)

6.4.2. Jpeg Module

Provides functionality for loading and saving images in JPEG format.

Header file: `Jpeg.h`.

Functions

JpegLoadImageFromFile	Loads image from JPEG file.
JpegLoadImageFromMemory	Loads image from memory buffer containing JPEG file.
JpegSaveImageToFile	Saves image to file in JPEG format with specified bitrate.
JpegSaveImageToMemory	Saves image to memory buffer in JPEG format with specified bitrate.

Macros

<code>JPEG_DEFAULT_QUALITY</code>	Specifies default JPEG quality.
-----------------------------------	---------------------------------

See Also

[NImages Library](#)

6.4.2.1. JpegLoadImageFromFile Function

Loads image from JPEG file.

```
NResult N_API JpegLoadImageFromFile(
    const NChar * szFileName,
    HNImage * pHImage
);
```

Parameters

<i>szFileName</i>	[in] Points to string that specifies file name.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>szFileName</i> or <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file specified by <i>szFileName</i> is invalid.

See Also

[Jpeg Module](#) | [HNImage](#) | [JpegLoadImageFromMemory](#) | [JpegSaveImageToFile](#)

6.4.2.2. JpegLoadImageFromMemory Function

Loads image from memory buffer containing JPEG file.

```
NResult N_API JpegLoadImageFromMemory(
    const void * buffer,
    NSizeType bufferLength,
    HNImage * pHImage
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer.
<i>bufferLength</i>	[in] Length of memory buffer.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL and <i>bufferLength</i> is not equal to zero. - or - <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file contained in buffer specified by <i>buffer</i> is invalid.

See Also

[Jpeg Module](#) | [HNImage](#) | [JpegLoadImageFromFile](#) | [JpegSaveImageToMemory](#)

6.4.2.3. JpegSaveImageToFile Function

Saves image to file in JPEG format with specified bitrate.

```
NResult N_API JpegSaveImageToFile(
    HNImage hImage,
    NInt quality,
    const NChar * szFileName
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>quality</i>	[in] Specifies quality of JPEG image.
<i>szFileName</i>	[in] Points to string that specifies file name.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>szFileName</i> is NULL .

See Also

[Jpeg Module](#) | [HNImage](#) | [JPEG_DEFAULT_QUALITY](#) | [JpegSaveImageToMemory](#) | [JpegLoadImageFromFile](#)

6.4.2.4. JpegSaveImageToMemory Function

Saves image to memory buffer in JPEG format with specified bitrate.

```
NResult N_API JpegSaveImageToMemory(
    HNImage hImage,
    NInt quality,
    void * * pBuffer,
    NSizeType * pBufferLength
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>quality</i>	[in] Specifies quality of JPEG image.
<i>pBuffer</i>	[out] Pointer to void * that receives pointer to allocated memory buffer.
<i>pBufferLength</i>	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> , <i>pBuffer</i> or <i>pBufferLength</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory to allocate memory buffer.

Remarks

Memory buffer allocated by the function must be deallocated using [NFree](#) function when it is no longer needed.

See Also

[Jpeg Module](#) | [HNImage](#) | [JPEG_DEFAULT_QUALITY](#) | [JpegSaveImageToFile](#) | [JpegLoadImageFromMemory](#)

6.4.3. NGrayscaleImage Module

Provides functionality for managing 8-bit grayscale images.

Header file: `NGrayscaleImage.h`.

Functions

NGrayscaleImageGetPixel	Retrieves value of pixel at the specified coordinates in 8-bit grayscale image.
NGrayscaleImageSetPixel	Sets value of pixel at the specified coordinates in 8-bit grayscale image.

Remarks

This module provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to [npfGrayscale](#).

See Also

[NImages Library](#) | [NImage Module](#)

6.4.3.1. NGrayscaleImageGetPixel Function

Retrieves value of pixel at the specified coordinates in 8-bit grayscale image.

```
NResult N_API NGrayscaleImageGetPixel(
    HNImage hImage,
    NUInt x,
    NUInt y,
    NByte * pValue
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>x</i>	[in] Specifies x-coordinate of the pixel.

<i>y</i>	[in] Specifies y-coordinate of the pixel.
<i>pValue</i>	[out] Points to NByte that receives pixel value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>x</i> is greater than or equal to image width. - or - <i>y</i> is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to npfGrayscale .

See Also

[NGrayscaleImage Module](#) | [HNImage](#) | [NGrayscaleImageSetPixel](#)

6.4.3.2. NGrayscaleImageSetPixel Function

Sets value of pixel at the specified coordinates in 8-bit grayscale image.

```
NResult N_API NGrayscaleImageSetPixel(
    HNImage hImage,
    NUInt x,
    NUInt y,
    NByte value
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>x</i>	[in] Specifies x-coordinate of the pixel.
<i>y</i>	[in] Specifies y-coordinate of the pixel.
<i>value</i>	[in] Specifies new pixel value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	

Error Code	Condition
	x is greater than or equal to image width. - or - y is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to npfGrayscale .

See Also

[NGrayscaleImage Module](#) | [HNImage](#) | [NGrayscaleImageGetPixel](#)

6.4.4. NImageFormat Module

Provides functionality for loading and saving images in format-neutral way.

Header file: `NImageFormat.h`.

Functions

NImageFormatCanRead	Retrieves a value indicating whether the image format supports reading.
NImageFormatCanWrite	Retrieves a value indicating whether the image format supports writing.
NImageFormatGetBmp	Retrieves BMP image format.
NImageFormatGetDefaultFileExtension	Retrieves default file extension of the image format.
NImageFormatGetFileFilter	Retrieves file filter of the image format.
NImageFormatGetFormat	Retrieves supported image format with specified index.
NImageFormatGetFormatCount	Retrieves number of supported image formats.
NImageFormatGetName	Retrieves name of the image format.
NImageFormatGetPng	Retrieves PNG image format.
NImageFormatGetTiff	Retrieves TIFF image format.
NImageFormatLoadImageFromFile	Loads image from file of specified image format.
NImageFormatLoadImageFromMemory	Loads image from the memory buffer containing file of specified image format.
NImageFormatSaveImageToFile	Saves image to the file in specified format.
NImageFormatSaveImageToMemory	Saves image to the memory buffer in specified format.
NImageFormatSelect	Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.

Types

HNImageFormat	Handle to image format.
-------------------------------	-------------------------

See Also

[NImages Library](#) | [NImage Module](#)

6.4.4.1. NImageFormatCanRead Function

Retrieves a value indicating whether the image format supports reading.

```
NResult N_API NImageFormatCanRead(
    HImageFormat hImageFormat,
    NBool * pValue
);
```

Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>pValue</i>	[out] Pointer to NBool that receives value indicating whether the image format supports reading.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> or <i>pValue</i> is NULL .

See Also

[NImageFormat Module](#) | [HImageFormat](#) | [NImageFormatCanWrite](#)

6.4.4.2. NImageFormatCanWrite Function

Retrieves a value indicating whether the image format supports writing.

```
NResult N_API NImageFormatCanWrite(
    HImageFormat hImageFormat,
    NBool * pValue
);
```

Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>pValue</i>	[out] Pointer to NBool that receives value indicating whether the image format supports writing.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> or <i>pValue</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatCanRead](#)

6.4.4.3. NImageFormatGetBmp Function

Retrieves BMP image format.

```
NResult N_API NImageFormatGetBmp(
    HNImageFormat * pValue
);
```

Parameters

<i>pValue</i>	[out] Pointer to HNImageFormat that receives handle to image format.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatGetTiff](#)

6.4.4.4. NImageFormatGetDefaultFileExtension Function

Retrieves default file extension of the image format.

```
NResult N_API NImageFormatGetDefaultFileExtension(
    HNImageFormat hImageFormat,
    NChar * pValue
);
```

Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>pValue</i>	[out] Pointer to string that receives default file extension of the image format. Can be NULL .

Return Values

If the function succeeds and *pValue* is [NULL](#), the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not [NULL](#), the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#)

6.4.4.5. NImageFormatGetFileFilter Function

Retrieves file filter of the image format.

```
NResult N_API NImageFormatGetFileFilter(
    HNImageFormat hImageFormat,
    NChar * pValue
);
```

Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>pValue</i>	[out] Pointer to string that receives file filter of the image format. Can be NULL .

Return Values

If the function succeeds and *pValue* is [NULL](#), the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not [NULL](#), the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#)

6.4.4.6. NImageFormatGetFormat Function

Retrieves supported image format with specified index.

```
NResult N_API NImageFormatGetFormat(
    NInt index,
    HNImageFormat * pValue
);
```

Parameters

<i>index</i>	[in] Specifies zero-based supported image format index to retrieve.
<i>pValue</i>	[out] Pointer to NImageFormat that receives image

	format.
--	---------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to supported image format count. See NImageFormatGetFormatCount .

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatGetFormatCount](#)

6.4.4.7. NImageFormatGetFormatCount Function

Retrieves number of supported image formats.

```
NResult N_API NImageFormatGetFormatCount(
    NInt * pValue
);
```

Parameters

<i>pValue</i>	[out] Pointer to NInt that receives number of supported image formats.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatGetFormat](#)

6.4.4.8. NImageFormatGetJpeg Function

Retrieves JPEG image format.

```
NResult N_API NImageFormatGetJpeg(
    HNImageFormat * pValue
);
```

Parameters

<i>pValue</i>	[out] Pointer to HNImageFormat that receives handle to image format.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatGetBmp](#)

6.4.4.9. NImageFormatGetName Function

Retrieves name of the image format.

```
NResult N_API NImageFormatGetName(
    HNImageFormat hImageFormat,
    NChar * pValue
);
```

Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>pValue</i>	[out] Pointer to string that receives name of the image format. Can be NULL .

Return Values

If the function succeeds and *pValue* is [NULL](#), the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not [NULL](#), the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#)

6.4.4.10. NImageFormatGetPng Function

Retrieves PNG image format.

```
NResult N_API NImageFormatGetPng(
    HNImageFormat * pValue
);
```

Parameters

<i>pValue</i>	[out] Pointer to HNImageFormat that receives handle to image format.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatGetBmp](#)

6.4.4.11. NImageFormatGetTiff Function

Retrieves TIFF image format.

```
NResult N_API NImageFormatGetTiff(
    HNImageFormat * pValue
);
```

Parameters

<i>pValue</i>	[out] Pointer to HNImageFormat that receives handle to image format.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatGetBmp](#)

6.4.4.12. NImageFormatLoadImageFromFile Function

Loads image from file of specified image format.

```
NResult N_API NImageFormatLoadImageFromFile(
    HNImageFormat hImageFormat,
    const NChar * szFileName,
    HNImage * pHImage
);
```


Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>szFileName</i>	[in] Points to string that specifies file name.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> , <i>szFileName</i> or <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file specified by <i>szFileName</i> is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by <i>hImageFormat</i> does not support reading.

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatCanRead](#) | [HNImage](#) | [NImageFormatLoadImageFromMemory](#) | [NImageFormatSaveImageToFile](#)

6.4.4.13. NImageFormatLoadImageFromMemory Function

Loads image from the memory buffer containing file of specified image format.

```
NResult N_API NImageFormatLoadImageFromMemory(
    HNImageFormat hImageFormat,
    void * buffer,
    NSizeType bufferLength,
    HNImage * pHImage
);
```

Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>buffer</i>	[in] Pointer to memory buffer.
<i>bufferLength</i>	[in] Length of memory buffer.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> or <i>pHImage</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferLength</i> is not equal to zero.
N_E_FORMAT	Format of file contained in buffer specified by <i>buffer</i> is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by <i>hImageFormat</i> does not support reading.

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatCanRead](#) | [HNImage](#) | [NImageFormatLoadImageFromFile](#) | [NImageFormatSaveImageToMemory](#)

6.4.4.14. NImageFormatSaveImageToFile Function

Saves image to the file in specified format.

```
NResult N_API NImageFormatSaveImageToFile(
    HNImageFormat hImageFormat,
    HNImage hImage,
    const NChar * szFileName
);
```

Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>hImage</i>	[in] Handle to image.
<i>szFileName</i>	[in] Points to string that specifies file name.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> , <i>hImage</i> or <i>szFileName</i> is NULL .
N_E_NOT_SUPPORTED	Image format specified by <i>hImageFormat</i> does not support writing.

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NImageFormatCanWrite](#) | [HNImage](#) | [NImageFormatSaveImageToMemory](#) | [NImageFormatLoadImageFromFile](#)

6.4.4.15. NImageFormatSaveImageToMemory Function

Saves image to the memory buffer in specified format.

```
NResult N_API NImageFormatSaveImageToMemory(
    HImageFormat hImageFormat,
    HImage hImage,
    void * * pBuffer,
    NSizeType * pBufferLength
);
```

Parameters

<i>hImageFormat</i>	[in] Handle to image format.
<i>hImage</i>	[in] Handle to image.
<i>pBuffer</i>	[out] Pointer to void * that receives pointer to allocated memory buffer.
<i>pBufferLength</i>	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImageFormat</i> , <i>hImage</i> , <i>pBuffer</i> or <i>pBufferLength</i> is NULL .
N_E_NOT_SUPPORTED	Image format specified by <i>hImageFormat</i> does not support writing.

Remarks

Memory buffer allocated by the function must be deallocated using [NFree](#) function when it is no longer needed.

See Also

[NImageFormat Module](#) | [HImageFormat](#) | [NImageFormatCanWrite](#) | [HImage](#) | [NImageFormatSaveImageToFile](#) | [NImageFormatLoadImageFromMemory](#)

6.4.4.16. NImageFormatSelect Function

Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.

```
NResult N_API NImageFormatSelect(
    const NChar * szFileName,
    NFileAccess fileAccess,
    HImageFormat * pHImageFormat
);
```

Parameters

<i>szFileName</i>	[in] Points to string that file name.
-------------------	---------------------------------------

<i>fileAccess</i>	[in] Specifies that image format should support reading, writing or both.
<i>pHImageFormat</i>	[out] Pointer to HNImageFormat that receives handle to image format.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>fileAccess</i> value is invalid.
N_E_ARGUMENT_NULL	<i>szFileName</i> or <i>pHImageFormat</i> is NULL .

Remarks

If none of supported image formats that supports reading/writing as specified by *fileAccess* is registered with file extension of *szFileName* then handle returned via *pHImageFormat* is [NULL](#).

See Also

[NImageFormat Module](#) | [HNImageFormat](#) | [NFileAccess](#) | [NImageFormatGetFormatCount](#) | [NImageFormatGetFormat](#)

6.4.5. NImage Module

Provides functionality for managing images.

Header file: `NImage.h`.

Functions

NImageClone	Creates a new image that is a copy of specified image.
NImageCreate	Creates an image with specified pixel format, size, stride and resolution.
NImageCreateFromData	Creates an image with specified pixel format, size, stride and resolution and copies specified pixels to it.
NImageCreateFromFile	Creates (loads) an image from file of specified format.
NImageCreateFromImage	Creates an image from specified image with specified pixel format and stride.
NImageCreateFromImageEx	Creates an image from specified image with specified pixel format, stride and resolution.
NImageCreateWrapper	Creates an image wrapper for specified pixels with specified pixel format, size, stride and resolution.
NImageFree	Deletes the image. After the image is deleted the specified handle is no longer valid.
NImageGetHeight	Retrieves height of the image.
NImageGetHorzResolution	Retrieves horizontal resolution of the image.
NImageGetPixelFormat	Retrieves pixel format of the image.

NImageGetPixels	Retrieves pointer to memory block containing pixels of the image.
NImageGetSize	Retrieves size of memory block containing pixels of the image.
NImageGetStride	Retrieves stride (size of one row) of the image.
NImageGetVertResolution	Retrieves vertical resolution of the image.
NImageGetWidth	Retrieves width of the image.
NImageSaveToFile	Saves the image to the file of specified format.

Types

HNImage	Handle to image.
-------------------------	------------------

See Also

[NImages Library](#) | [NMonochromeImage Module](#) | [NGrayscaleImage Module](#) | [NRGBImage Module](#) | [NImageFormat Module](#)

6.4.5.1. NImageClone Function

Creates a new image that is a copy of specified image.

```
NResult N_API NImageClone(
    HNImage hImage,
    HNImage * pHClonedImage
);
```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pHClonedImage</i>	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pHClonedImage</i> is NULL .

Remarks

Created image must be deleted using [NImageFree](#) function.

See Also

[NImage Module](#) | [HNImage](#) | [NImageFree](#) | [NImageCreate](#)

6.4.5.2. NImageCreate Function

Creates an image with specified pixel format, size, stride and resolution.

```
NResult N_API NImageCreate(
    NPixelFormat pixelFormat,
    NUInt width,
    NUInt height,
    NSizeType stride,
    NFloat horzResolution,
    NFloat vertResolution,
    HNImage * pHImage
);
```

Parameters

<i>pixelFormat</i>	[in] Specifies pixel format of the image.
<i>width</i>	[in] Specifies width of the image.
<i>height</i>	[in] Specifies height of the image.
<i>stride</i>	[in] Specifies stride of the image. Can be zero.
<i>horzResolution</i>	[in] Specifies horizontal resolution in pixels per inch of the image.
<i>vertResolution</i>	[in] Specifies vertical resolution in pixels per inch of the image.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>pixelFormat</i> has invalid value. - or - <i>stride</i> is not zero and is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	<i>pHImage</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is less than zero.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see [NImage-GetStride](#) function.

Created image must be deleted using [NImageFree](#) function.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

[NImage Module](#) | [HNImage](#) | [NImageFree](#) | [NImageCreateWrapper](#) | [NImageCreateFromData](#) | [NImageCreateFromImage](#) | [NImageCreateFromFile](#) | [NImageClone](#) | [NImageGetStride](#)

6.4.5.3. NImageCreateFromData Function

Creates an image with specified pixel format, size, stride and resolution and copies specified pixels to it.

```
NResult N_API NImageCreateFromData(
    NPixelFormat pixelFormat,
    NUInt width,
    NUInt height,
    NSizeType stride,
    NFloat horzResolution,
    NFloat vertResolution,
    NSizeType srcStride,
    const void * srcPixels,
    HNImage * pHImage
);
```

Parameters

<i>pixelFormat</i>	[in] Specifies pixel format of the image.
<i>width</i>	[in] Specifies width of the image.
<i>height</i>	[in] Specifies height of the image.
<i>stride</i>	[in] Specifies stride of the image. Can be zero.
<i>horzResolution</i>	[in] Specifies horizontal resolution in pixels per inch of the image.
<i>vertResolution</i>	[in] Specifies vertical resolution in pixels per inch of the image.
<i>srcStride</i>	[in] Specifies stride of pixels to be copied to the image.
<i>srcPixels</i>	[in] Points to memory block containing pixels that to be copied to the image.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<p><i>pixelFormat</i> has invalid value.</p> <p>- or -</p> <p><i>stride</i> is not zero and is less than minimal value for specified pixel format and width.</p>

Error Code	Condition
	- or - <i>srcStride</i> is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	<i>srcPixels</i> or <i>pHImage</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is less than zero.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see [NImage-GetStride](#) function.

Format of memory block *srcPixels* points to must be the same as described in [NImageGetPixels](#) function, only stride is equal to *srcStride*.

Created image must be deleted using [NImageFree](#) function.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

[NImage Module](#) | [HNImage](#) | [NImageFree](#) | [NImageCreate](#) | [NImageCreateWrapper](#) | [NImageGetStride](#) | [NImageGetPixels](#)

6.4.5.4. NImageCreateFromFile Function

Creates (loads) an image from file of specified format.

```
NResult N_API NImageCreateFromFile(
    const NChar * szFileName,
    HNImageFormat hImageFormat,
    HNImage * pHImage
);
```

Parameters

<i>szFileName</i>	[in] Points to string that specifies file name.
<i>hImageFormat</i>	[in] Handle to the image format of the file. Can be NULL .
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>szFileName</i> or <i>pHImage</i> is NULL .

Error Code	Condition
N_E_FORMAT	Format of file specified by <i>szFileName</i> is invalid for specified image format.
N_E_NOT_SUPPORTED	<p><i>hImageFormat</i> is NULL and none of supported image formats is registered with file extension of <i>szFileName</i>.</p> <p>- or -</p> <p><i>hImageFormat</i> is NULL and image format registered with file extension of <i>szFileName</i> does not support reading.</p> <p>- or -</p> <p>Image format specified by <i>hImageFormat</i> does not support reading.</p>

Remarks

If *hImageFormat* is [NULL](#) image format is selected by file extension of *szFileName*.

Created image must be deleted using [NImageFree](#) function.

See Also

[NImage Module](#) | [HNImage](#) | [NImageFree](#) | [NImageCreate](#) | [NImageFormatCanRead](#)

6.4.5.5. NImageCreateFromImage Function

Creates an image from specified image with specified pixel format and stride.

```
NResult N_API NImageCreateFromImage(
    NPixelFormat pixelFormat,
    NSizeType stride,
    HNImage hSrcImage,
    HNImage * pHImage
);
```

Parameters

<i>pixelFormat</i>	[in] Specifies pixel format of the image.
<i>stride</i>	[in] Specifies stride of the image. Can be zero.
<i>hSrcImage</i>	[in] Handle to image used as source for the image.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>pixelFormat</i> has invalid value.

Error Code	Condition
	- or - <i>stride</i> is not zero and is less than minimal value for specified pixel format and source image width.
N_E_ARGUMENT_NULL	<i>hSrcImage</i> or <i>pHImage</i> is NULL .

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see [NImage-GetStride](#) function.

Created image must be deleted using [NImageFree](#) function.

See Also

[NImage Module](#) | [HNImage](#) | [NImageFree](#) | [NImageCreate](#) | [NImageCreateFromImageEx](#) | [NImageClone](#) | [NImageGetStride](#)

6.4.5.6. NImageCreateFromImageEx Function

Creates an image from specified image with specified pixel format, stride and resolution.

```
NResult N_API NImageCreateFromImageEx(
    NPixelFormat pixelFormat,
    NSizeType stride,
    NFloat horzResolution,
    NFloat vertResolution,
    HNImage hSrcImage,
    HNImage * pHImage
);
```

Parameters

<i>pixelFormat</i>	[in] Specifies pixel format of the image.
<i>stride</i>	[in] Specifies stride of the image. Can be zero.
<i>horzResolution</i>	[in] Specifies horizontal resolution in pixels per inch of the image.
<i>vertResolution</i>	[in] Specifies vertical resolution in pixels per inch of the image.
<i>hSrcImage</i>	[in] Handle to image used as source for the image.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>pixelFormat</i> has invalid value. - or -

Error Code	Condition
	<i>stride</i> is not zero and is less than minimal value for specified pixel format and source image width.
N_E_ARGUMENT_NULL	<i>hSrcImage</i> or <i>pHImage</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>horzResolution</i> or <i>vertResolution</i> is less than zero.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see [NImageGetStride](#) function.

Created image must be deleted using [NImageFree](#) function.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

[NImage Module](#) | [HNImage](#) | [NImageFree](#) | [NImageCreate](#) | [NImageCreateFromImage](#) | [NImageClone](#) | [NImageGetStride](#)

6.4.5.7. NImageCreateWrapper Function

Creates an image wrapper for specified pixels with specified pixel format, size, stride and resolution.

```
NResult N_API NImageCreateWrapper(
    NPixelFormat pixelFormat,
    NUInt width,
    NUInt height,
    NSizeType stride,
    NFloat horzResolution,
    NFloat vertResolution,
    void * pixels,
    NBool ownsPixels,
    HNImage * pHImage
);
```

Parameters

<i>pixelFormat</i>	[in] Specifies pixel format of the image.
<i>width</i>	[in] Specifies width of the image.
<i>height</i>	[in] Specifies height of the image.
<i>stride</i>	[in] Specifies stride of the image.
<i>horzResolution</i>	[in] Specifies horizontal resolution in pixels per inch of the image.
<i>vertResolution</i>	[in] Specifies vertical resolution in pixels per inch of the image.
<i>pixels</i>	[in] Points to memory block containing pixels for the image.
<i>ownsPixels</i>	[in] Specifies whether pixels will be automatically deleted with the image (if set to NTrue).

<i>pHImage</i>	[out] Pointer to HNImage that receives handle to created image.
----------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>pixelFormat</i> has invalid value. - or - <i>stride</i> is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	<i>pixels</i> or <i>pHImage</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is less than zero.

Remarks

For more information on image stride see [NImageGetStride](#) function.

Format of memory block *pixels* points to must be the same as described in [NImageGetPixels](#) function.

Created image must be deleted using [NImageFree](#) function.

pixels must not be deleted during lifetime of the image. If *ownsPixels* is [NTrue](#) then *pixels* will be automatically deleted with the image.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

[NImage Module](#) | [HNImage](#) | [NImageFree](#) | [NImageCreate](#) | [NImageCreateFromData](#) | [NImageGetStride](#) | [NImageGetPixels](#)

6.4.5.8. NImageFree Function

Deletes the image. After the image is deleted the specified handle is no longer valid.

```
void N\_API NImageFree(
    HNImage hImage
);
```

Parameters

<i>hImage</i>	[in] Handle to the image.
---------------	---------------------------

Remarks

If *hImage* is [NULL](#) does nothing.

See Also

[NImage Module](#) | [HNImage](#) | [NImageCreate](#)

6.4.5.9. NImageGetHeight Function

Retrieves height of the image.

```
NResult N_API NImageGetHeight(  
    HNImage hImage,  
    NUInt * pValue  
);
```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pValue</i>	[out] Pointer to NUInt that receives height of the image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .

See Also

[NImage Module](#) | [HNImage](#) | [NImageGetWidth](#)

6.4.5.10. NImageGetHorzResolution Function

Retrieves horizontal resolution of the image.

```
NResult N_API NImageGetHorzResolution(  
    HNImage hImage,  
    NFloat * pValue  
);
```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pValue</i>	[out] Pointer to NFloat that receives horizontal resolution in pixels per inch of the image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .

Remarks

Horizontal resolution equal to zero means that it is not applicable for the image.

See Also

[NImage Module](#) | [HNImage](#) | [NImageGetVertResolution](#)

6.4.5.11. NImageGetPixelFormat Function

Retrieves pixel format of the image.

```
NResult N_API NImageGetPixelFormat(
    HNImage hImage,
    NPixelFormat * pValue
);
```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pValue</i>	[out] Pointer to NPixelFormat that receives pixel format of the image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .

See Also

[NImage Module](#) | [HNImage](#) | [NPixelFormat](#)

6.4.5.12. NImageGetPixels Function

Retrieves pointer to memory block containing pixels of the image.

```
NResult N_API NImageGetPixels(
    HNImage hImage,
    void * * pValue
);
```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pValue</i>	[out] Pointer to void * that receives pointer to memory block containing pixels of the image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .

Remarks

Memory block containing image pixels is organized as image height rows following each other in top-to-bottom order. Each row occupies image stride bytes and is organized as image width pixels following each other in right-to-left order. Each pixel is described by image pixel format.

For more information see [NImageGetPixelFormat](#), [NImageGetWidth](#), [NImageGetHeight](#), [NImageGetStride](#), and [NImageGetSize](#) functions.

See Also

[NImage Module](#) | [HNImage](#) | [NImageGetPixelFormat](#) | [NImageGetWidth](#) | [NImageGetHeight](#) | [NImageGetStride](#) | [NImageGetSize](#)

6.4.5.13. NImageGetSize Function

Retrieves size of memory block containing pixels of the image.

```
NResult N_API NImageGetSize(
    HNImage hImage,
    NSizeType * pValue
);
```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pValue</i>	[out] Pointer to NSizeType that receives size of memory block containing pixels of the image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .

Remarks

Size of memory block containing image pixels is equal to image height multiplied by image stride. For more information see [NImageGetHeight](#) and [NImageGetStride](#) functions.

See Also

[NImage Module](#) | [HNImage](#) | [NImageGetHeight](#) | [NImageGetStride](#)

6.4.5.14. NImageGetStride Function

Retrieves stride (size of one row) of the image.

```
NResult N_API NImageGetStride(
```

```

    HNIImage hImage,
    NSizeType * pValue
);

```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pValue</i>	[out] Pointer to NSizeType that receives stride of the image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .

Remarks

Stride (size of one row) of the image depends on image pixel format and width. It cannot be less than value obtained with [NPixelFormatGetRowSize](#) macro with arguments obtained with [NImageGetPixelFormat](#) and [NImageGetWidth](#) functions.

See Also

[NImage Module](#) | [HNIImage](#) | [NPixelFormatGetRowSize](#) | [NImageGetPixelFormat](#) | [NImageGetWidth](#) | [NImageGetSize](#)

6.4.5.15. NImageGetVertResolution Function

Retrieves vertical resolution of the image.

```

NResult N_API NImageGetVertResolution(
    HNIImage hImage,
    NFloat * pValue
);

```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pValue</i>	[out] Pointer to NFloat that receives vertical resolution in pixels per inch of the image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .

Remarks

Vertical resolution equal to zero means that it is not applicable for the image.

See Also

[NImage Module](#) | [HNImage](#) | [NImageGetHorzResolution](#)

6.4.5.16. NImageGetWidth Function

Retrieves width of the image.

```
NResult N_API NImageGetWidth(
    HNImage hImage,
    NUInt * pValue
);
```

Parameters

<i>hImage</i>	[in] Handle to the image.
<i>pValue</i>	[out] Pointer to NUInt that receives width of the image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .

See Also

[NImage Module](#) | [HNImage](#) | [NImageGetHeight](#) | [NImageGetStride](#)

6.4.5.17. NImageSaveToFile Function

Saves the image to the file of specified format.

```
NResult N_API NImageSaveToFile(
    HNImage hImage,
    const NChar * szFileName,
    HNImageFormat hImageFormat
);
```

Parameters

<i>hImage</i>	[in] Handle to NImage object.
<i>szFileName</i>	[in] Points to string that specifies file name.
<i>hImageFormat</i>	[in] Handle to the image format of the file. Can be NULL .

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>szFileName</i> is NULL .
N_E_NOT_SUPPORTED	<p><i>hImageFormat</i> is NULL and none of supported image formats is registered with file extension of <i>szFileName</i>.</p> <p>- or -</p> <p><i>hImageFormat</i> is NULL and image format registered with file extension of <i>szFileName</i> does not support writing.</p> <p>- or -</p> <p>Image format specified by <i>hImageFormat</i> does not support writing.</p>

Remarks

If *hImageFormat* is [NULL](#) image format is selected by file extension of *szFileName*.

See Also

[NImage Module](#) | [HNImage](#) | [NImageCreateFromFile](#) | [NImageFormatCanWrite](#)

6.4.6. Nimages Module

Provides library registration and other additional functionality.

Header file: `NImages.h`.

Functions

NImagesGetGrayscaleColorWrapper	Creates color wrapper for grayscale image.
---	--

See Also

[NImages Library](#)

6.4.6.1. NImagesGetGrayscaleColorWrapper Function

Creates color wrapper for grayscale image.

```
NResult N_API NImagesGetGrayscaleColorWrapper(
    HNImage hImage,
    NRgb minColor,
    NRgb maxColor,
    HNImage * pHDstImage
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>minColor</i>	[in] Specifies color to be used for black color.

<i>maxColor</i>	[in] Specifies color to be used for white color.
<i>pHDstImage</i>	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Image specified by <i>hImage</i> has non-grayscale pixel format (not npfGrayscale or npfMonochrome).
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pHDstImage</i> is NULL .

Remarks

Created image must be deleted using [NImageFree](#) function.

Created image is a thin wrapper for specified grayscale image. Therefore *hImage* must not be freed before created image.

Gray values in source image are replaced with according RGB values from range [*minColor*, *maxColor*] in created image.

See Also

[NImages Module](#) | [HNImage](#) | [NImageFree](#)

6.4.7. NMonochromeImage Module

Provides functionality for managing 1-bit monochrome images.

Header file: `NMonochromeImage.h`.

Functions

NMonochromeImageGetPixel	Retrieves value of pixel at the specified coordinates in 1-bit monochrome image.
NMonochromeImageSetPixel	Sets value of pixel at the specified coordinates in 1-bit monochrome image.

Remarks

This module provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to [npfMonochrome](#).

See Also

[NImages Library](#) | [NImage Module](#)

6.4.7.1. NMonochromeImageGetPixel Function

Retrieves value of pixel at the specified coordinates in 1-bit monochrome image.

```
NResult N_API NMonochromeImageGetPixel(
    HNImage hImage,
```

```

    NUInt x,
    NUInt y,
    NBool * pValue
);

```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>x</i>	[in] Specifies x-coordinate of the pixel.
<i>y</i>	[in] Specifies y-coordinate of the pixel.
<i>pValue</i>	[out] Points to NBool that receives pixel value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>x</i> is greater than or equal to image width. - or - <i>y</i> is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to npfMonochrome .

Remarks

If pixel is black then value *pValue* points to receives [NFalse](#) and if it is white then value receives [NTrue](#).

See Also

[NMonochromeImage Module](#) | [HNImage](#) | [NMonochromeImageSetPixel](#)

6.4.7.2. NMonochromeImageSetPixel Function

Sets value of pixel at the specified coordinates in 1-bit monochrome image.

```

NResult N_API NMonochromeImageSetPixel(
    HNImage hImage,
    NUInt x,
    NUInt y,
    NBool value
);

```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>x</i>	[in] Specifies x-coordinate of the pixel.
<i>y</i>	[in] Specifies y-coordinate of the pixel.
<i>value</i>	[in] Specifies new pixel value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>x</i> is greater than or equal to image width. - or - <i>y</i> is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to npfMonochrome .

Remarks

If *value* is [NFalse](#) then pixel will be black and if it is [NTrue](#) then pixel will be white.

See Also

[NMonochromeImage Module](#) | [HNImage](#) | [NMonochromeImageGetPixel](#)

6.4.8. NPixelFormat Module

Provides functionality for working with image pixel format.

Header file: `NPixelFormat.h`.

Functions

<code>NPixelFormatGetBitsPerPixelFunc</code>	Used internally in NPixelFormatGetBitsPerPixel macro.
<code>NPixelFormatIsValid</code>	Checks if specified pixel format is valid.

Structures

NRGB	Represents an RGB color.
----------------------	--------------------------

Enumerations

NPixelFormat	Specifies pixel format of each pixel in the image.
------------------------------	--

Macros

<code>NCalcRowSize</code>	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
<code>NCalcRowSizeEx</code>	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel and alignment.
<code>NPixelFormatGetBitsPerPixel</code>	Retrieves number of bits used to store a pixel from NPixelFormat .
<code>NPixelFormatGetRowSize</code>	Calculates number of bytes needed to store line of spe-

	cified length of pixels with specified NPixelFormat .
<code>NPixelFormatGetRowSizeEx</code>	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat and alignment.
<code>NRgbConst</code>	Makes NRgb constant with field values provided.

See Also

[NImages Library](#)

6.4.8.1. NPixelFormat Enumeration

Specifies pixel format of each pixel in the image.

```
typedef enum NPixelFormat_ { } NPixelFormat;
```

Members

<code>npfGrayscale</code>	Each pixel value is stored in 8 bits representing 256 shades of gray.
<code>npfMonochrome</code>	Each pixel value is stored in 1 bit representing either black or white color.
<code>npfRgb</code>	Each pixel value is stored in 24 bits consisting of three 8-bit values representing red, green and blue color components.

Remarks

Image pixel format is not limited to members of this enumeration. However only these members are provided for usage with this product.

See Also

[NPixelFormat Module](#) | [HNImage](#)

6.4.8.2. NRgb Structure

Represents an RGB color.

```
typedef struct NRgb_ { } NRgb;
```

Fields

Blue	Blue component value of this NRgb .
Green	Green component value of this NRgb .
Red	Red component value of this NRgb .

See Also

[NPixelFormat Module](#)

6.4.8.2.1. NRgb.Blue Field

Blue component value of this [NRGB](#).

```
NByte Blue;
```

See Also

[NRGB Structure](#)

6.4.8.2.2. NRGB.Green Field

Green component value of this [NRGB](#).

```
NByte Green;
```

See Also

[NRGB Structure](#)

6.4.8.2.3. NRGB.Red Field

Red component value of this [NRGB](#).

```
NByte Red;
```

See Also

[NRGB Structure](#)

6.4.9. NRGBImage Module

Provides functionality for managing 24-bit RGB images.

Header file: `NRGBImage.h`.

Functions

NRGBImageGetPixel	Retrieves value of pixel at the specified coordinates in 24-bit RGB image.
NRGBImageSetPixel	Sets value of pixel at the specified coordinates in 24-bit RGB image.

Remarks

This module provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to [npfRGB](#).

See Also

[NImages Library](#) | [NImage Module](#)

6.4.9.1. NRGBImageGetPixel Function

Retrieves value of pixel at the specified coordinates in 24-bit RGB image.

```
NResult N_API NRGBImageGetPixel(
    HImage hImage,
    NUInt x,
    NUInt y,
    NRGB * pValue
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>x</i>	[in] Specifies x-coordinate of the pixel.
<i>y</i>	[in] Specifies y-coordinate of the pixel.
<i>pValue</i>	[out] Pointer to NRgb that receives pixel value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>x</i> is greater than or equal to image width. - or - <i>y</i> is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to npfRgb .

See Also

[NRgbImage Module](#) | [HNImage](#) | [NRgb](#) | [NRgbImageSetPixel](#)

6.4.9.2. NRgbImageSetPixel Function

Sets value of pixel at the specified coordinates in 24-bit RGB image.

```
NResult N_API NRgbImageSetPixel(
    HNImage hImage,
    NUInt x,
    NUInt y,
    const NRGB * pValue
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>x</i>	[in] Specifies x-coordinate of the pixel.
<i>y</i>	[in] Specifies y-coordinate of the pixel.
<i>pValue</i>	[in] Pointer to NRgb that specifies new pixel value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>x</i> is greater than or equal to image width. - or - <i>y</i> is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to npfRgb .

See Also

[NRgbImage Module](#) | [HNImage](#) | [NRgb](#) | [NRgbImageGetPixel](#)

6.4.10. Png Module

Provides functionality for loading and saving images in PNG format.

Header file: `Png.h`.

Functions

PngLoadImageFromFile	Loads image from PNG file.
PngLoadImageFromMemory	Loads image from the memory buffer containing PNG file.
PngLoadImageFromStream	Loads image from the stream containing PNG file.
PngSaveImageToFile	Saves image to file in PNG format.
PngSaveImageToMemory	Saves image to the memory buffer in PNG format.
PngSaveImageToStream	Saves image to the stream in PNG format.

Macros

<code>PNG_DEFAULT_COMPRESSION_LEVEL</code>	Specifies default PNG compression level.
--	--

See Also

[NImages Library](#)

6.4.10.1. PngLoadImageFromFile Function

Loads image from PNG file.

```
NResult N_API PngLoadImageFromFile(
    const NChar * szFileName,
    HNImage * pHImage
);
```

Parameters

<i>szFileName</i>	[in] Points to string that specifies file name.
<i>pHImage</i>	[out] Pointer to #HNImage that receives handle to loaded

	image.
--	--------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>szFileName</i> or <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file specified by <i>szFileName</i> is invalid.

Remark

This is a low-level function and can be changed in future version of the library.

See Also

[Png Module](#) | [HNImage](#) | [PngLoadImageFromMemory](#) | [PngLoadImageFromStream](#) | [PngSaveImageToFile](#)

6.4.10.2. PngLoadImageFromMemory Function

Loads image from the memory buffer containing PNG file.

```
NResult N_API PngLoadImageFromMemory(
    const void * buffer,
    NSizeType bufferSize,
    HNImage * pHImage
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer.
<i>bufferLength</i>	[in] Length of memory buffer.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL and <i>bufferLength</i> is not equal to zero or <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file contained in memory buffer specified by <i>buffer</i> is invalid.

Remark

This is a low-level function and can be changed in future version of the library.

See Also

[Png Module](#) | [HNImage](#) | [PngLoadImageFromFile](#) | [PngLoadImageFromStream](#) | [PngSaveImageToMemory](#)

6.4.10.3. PngLoadImageFromStream Function

Loads image from the stream containing PNG file.

```
NResult N_API PngLoadImageFromStream(
    HNStream hStream,
    HNImage * pHImage
);
```

Parameters

<i>hStream</i>	[in] Handle to HNStream.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL and <i>bufferLength</i> is not equal to zero or <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file contained in memory buffer specified by <i>buffer</i> is invalid.

Remark

This is a low-level function and can be changed in future version of the library.

See Also

[Png Module](#) | [HNImage](#) | [PngLoadImageFromFile](#) | [PngLoadImageFromMemory](#) | [PngSaveImageToMemory](#)

6.4.10.4. PngSaveImageToFile Function

Saves image to file in PNG format.

```
NResult N_API PngSaveImageToFile(
    HNImage hImage,
    NInt compressionLevel,
    const NChar * szFileName
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>compressionLevel</i>	[in] Specifies level of PNG compression. The value can be in range [0, 9]. 0 - uncompressed, 9 - maximal compression level. See PNG_DEFAULT_COMPRESSION_LEVEL

<i>szFileName</i>	[in] Points to string that specifies file name.
-------------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> or <i>szFileName</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>compressionLevel</i> is less than 0 or greater than 9.

Remark

This is a low-level function and can be changed in future version of the library.

See Also

[Png Module](#) | [HNImage](#) | [PngSaveImageToMemory](#) | [PngSaveImageToStream](#) | [PngLoadImageFromFile](#)

6.4.10.5. PngSaveImageToMemory Function

Saves image to the memory buffer in PNG format.

```
NResult N_API PngSaveImageToMemory(
    HNImage hImage,
    NInt compressionLevel,
    void * * pBuffer,
    NSizeType * pBufferLength
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>compressionLevel</i>	[in] Specifies level of PNG compression. The value can be in range [0, 9]. 0 - uncompressed, 9 - maximal compression level. See PNG_DEFAULT_COMPRESSION_LEVEL
<i>pBuffer</i>	[out] Pointer to void * that receives pointer to allocated memory buffer.
<i>pBufferLength</i>	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> , <i>pBuffer</i> or <i>pBufferLength</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory to allocate memory buffer.

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	<i>compressionLevel</i> is less than 0 or greater than 9.

Remark

This is a low-level function and can be changed in future version of the library.

Memory buffer allocated by this function must be deallocated by [NFree](#) function when it is no longer needed.

See Also

[Png Module](#) | [HNImage](#) | [PngSaveImageToFile](#) | [PngSaveImageToStream](#) | [PngLoadImageFromMemory](#)

6.4.10.6. PngSaveImageToStream Function

Saves image to the stream in PNG format.

```
NResult N_API PngSaveImageToStream(
    HNImage hImage,
    NInt compressionLevel,
    HNStream hStream
);
```

Parameters

<i>hImage</i>	[in] Handle to image.
<i>compressionLevel</i>	[in] Specifies level of PNG compression. The value can be in range [0, 9]. 0 - uncompressed, 9 - maximal compression level. See PNG_DEFAULT_COMPRESSION_LEVEL
<i>hStream</i>	[out] Handle to HNStream where image will be written.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hImage</i> , <i>pBuffer</i> or <i>pBufferLength</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory to allocate memory buffer.
N_E_ARGUMENT_OUT_OF_RANGE	<i>compressionLevel</i> is less than 0 or greater than 9.

Remark

This is a low-level function and can be changed in future version of the library.

Memory buffer allocated by this function must be deallocated by [NFree](#) function when it is no longer needed.

See Also

[Png Module](#) | [HNImage](#) | [PngSaveImageToFile](#) | [PngSaveImageToMemory](#) | [PngLoadImageFromStream](#)

6.4.11. Tiff Module

Provides functionality for loading images in TIFF format.

Header file: `Tiff.h`.

Functions

TiffLoadImageFromFile	Loads image from TIFF file.
TiffLoadImageFromMemory	Loads image from memory buffer containing TIFF file.

See Also

[NImages Library](#)

6.4.11.1. TiffLoadImageFromFile Function

Loads image from TIFF file.

```
NResult N_API TiffLoadImageFromFile(
    const NChar * szFileName,
    HNImage * pHImage
);
```

Parameters

<i>szFileName</i>	[in] Points to string that specifies file name.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>szFileName</i> or <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file specified by <i>szFileName</i> is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

[Tiff Module](#) | [HNImage](#) | [TiffLoadImageFromMemory](#)

6.4.11.2. TiffLoadImageFromMemory Function

Loads image from memory buffer containing TIFF file.

```
NResult N_API TiffLoadImageFromMemory(
    const void * buffer,
    NSizeType bufferLength,
    HNImage * pHImage
);
```

```
) ;
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer.
<i>bufferLength</i>	[in] Length of memory buffer.
<i>pHImage</i>	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL and <i>bufferLength</i> is not equal to zero. - or - <i>pHImage</i> is NULL .
N_E_FORMAT	Format of file contained in buffer specified by <i>buffer</i> is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

[Tiff Module](#) | [HNImage](#) | [TiffLoadImageFromFile](#)

6.5. NTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnology Templates (NTemplates), Fingers Templates (NFTemplates), Faces Templates (NLTemplates), Iris Templates (NETemplates), Finger Records (NFRecords), Face Records (NLRecords), Iris Records (NERecords).

Import library: `NTemplate.dll.lib`.

DLL: `NTemplate.dll`.

Requirements:

- [NCore.dll](#).

Linux

Shared object: `libNTemplate.so`.

Requirements:

- [libNCore.so](#).

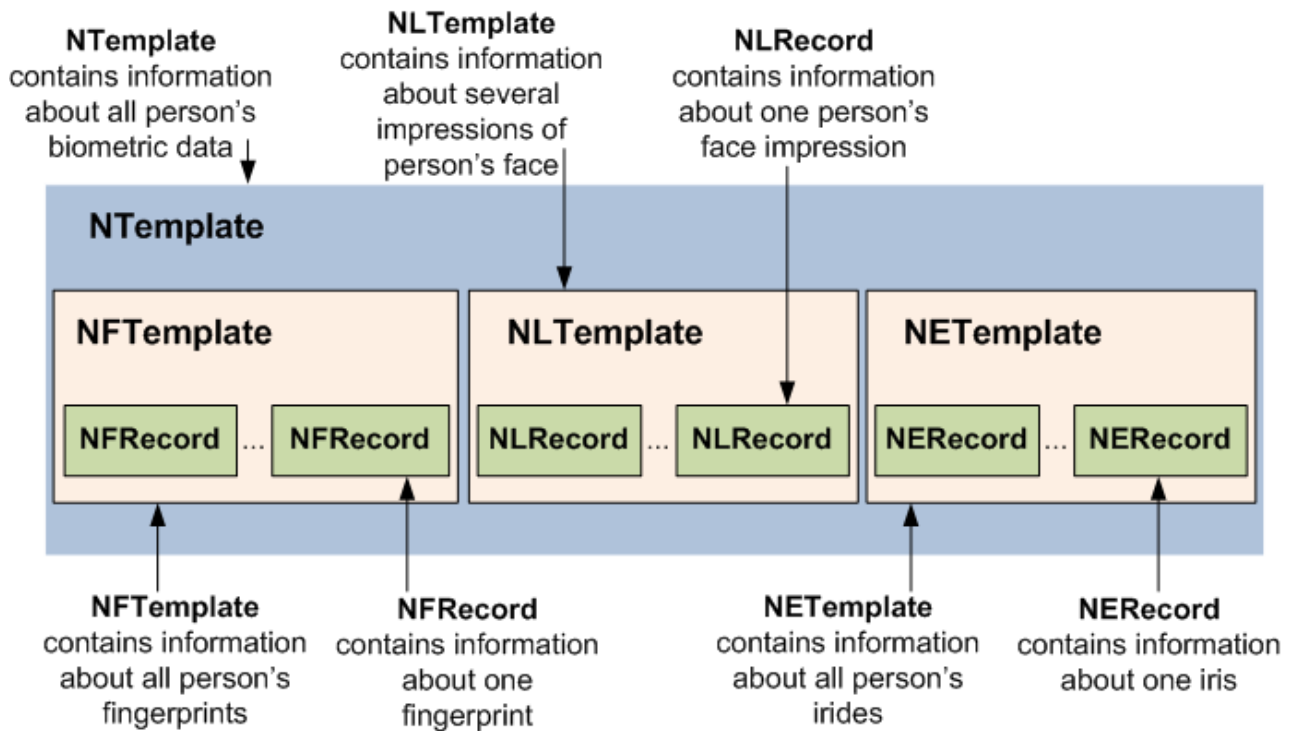


Figure 6.1. Structure of NTemplate.

Modules

NTemplate	Provides functionality for packing, unpacking and editing Neurotechnology Templates (NTemplates).
NFRecord	Provides functionality for packing, unpacking and editing Neurotechnology Finger Records (NFRecords).
NFTemplate	Provides functionality for packing, unpacking and editing Neurotechnology Fingers Templates (NFTemplates).
NLRecord	Provides functionality for packing, unpacking and editing Neurotechnology Face Records (NLRecords).
NLTemplate	Provides functionality for packing, unpacking and editing Neurotechnology Faces Templates (NLTemplates).
NETemplate	Provides functionality for packing, unpacking and editing Neurotechnology iris templates (NETemplates).
NERRecord	Provides functionality for packing, unpacking and editing Neurotechnology iris Records (NERRecords).

6.5.1. NTemplate Module

Provides functionality for packing, unpacking and editing Neurotechnology Templates (NTemplates).

Header file: `NTemplate.h`.

Functions

NTemplateAddFaces	Adds an empty faces template to the NTemplate.
NTemplateAddFacesCopy	Adds a copy of the faces template to the NFTemplate.

NTemplateAddFacesFromMemory	Unpacks a faces template from the specified memory buffer and adds it to the NTemplate.
NTemplateAddFingers	Adds an empty fingers template to the NTemplate.
NTemplateAddFingersCopy	Adds a copy of the fingers template to the NTemplate.
NTemplateAddFingersFromMemory	Unpacks a fingers template from the specified memory buffer and adds it to the NTemplate.
NTemplateCalculateSize	Calculates the size of a packed NTemplate containing fingers and faces templates of the specified size.
NTemplateCheck	Checks if format of the packed NTemplate is correct.
NTemplateClear	Removes all templates from the NTemplate.
NTemplateClone	Creates a copy of the NTemplate.
NTemplateCreate	Creates an empty NTemplate.
NTemplateCreateFromMemory	Unpacks a NTemplate from the specified memory buffer.
NTemplateFree	Deletes the NTemplate. After the object is deleted the specified handle is no longer valid.
NTemplateGetFaces	Retrieves the faces template of the NTemplate.
NTemplateGetFingers	Retrieves the fingers template of the NTemplate.
NTemplateGetSize	Calculates packed size of the NTemplate.
NTemplateInfoDispose	For internal use.
NTemplatePack	Packs packed fingers and faces templates as NTemplate into the specified memory buffer.
NTemplateRemoveFaces	Removes faces template from the NTemplate.
NTemplateRemoveFingers	Removes fingers template from the NTemplate.
NTemplateSaveToMemory	Packs the NTemplate into the specified memory buffer.
NTemplateUnpack	Unpacks packed fingers template from the packed NTemplate.

Structures

NTemplateInfo	For internal use.
---------------	-------------------

Types

HNTemplate	Handle to NTemplate object.
------------	-----------------------------

See Also

[NTemplate Library](#)

6.5.1.1. NTemplateAddFaces Function

Adds an empty fingers template to the NTemplate.

```
NResult N_API NTemplateAddFaces(
    HNTemplate hTemplate,
    * pHFaces
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>pHFaces</i>	[out] Pointer to a that receives handle to created NLTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHFaces</i> is NULL .
N_E_INVALID_OPERATION	NTemplate already contains faces template. See NTemplateGetFaces function.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [NTemplateGetFaces](#)

6.5.1.2. NTemplateAddFacesCopy Function

Adds a copy of the faces template to the NTemplate.

```
NResult N_API NTemplateAddFacesCopy(
    HNTemplate hTemplate,
    hSrcFaces,
    * pHFaces
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>hSrcFaces</i>	[in] Handle to the NLTemplate object.
<i>pHFaces</i>	[out] Pointer to a that receives handle to created NLTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>hSrcFaces</i> or <i>pHFaces</i> is NULL .

Error Code	Condition
N_E_INVALID_OPERATION	NTemplate already contains faces template. See NTemplateGetFaces function.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [NTemplateGetFaces](#)

6.5.1.3. NTemplateAddFacesFromMemory Function

Unpacks a faces template from the specified memory buffer and adds it to the NLTemplate.

```
NResult N_API NTemplateAddFacesFromMemory(
    HNTemplate hTemplate,
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    * pHFaces
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLTemplate.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pHFaces</i>	[out] Pointer to that receives handle to created NLTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHFaces</i> , or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLTemplate format.
N_E_INVALID_OPERATION	NTemplate already contains faces template. See NTemplateGetFaces function.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see function.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [NTemplateGetFaces](#)

6.5.1.4. NTemplateAddFingers Function

Adds an empty fingers template to the NTemplate.

```
NResult N_API NTemplateAddFingers(
    HNTemplate hTemplate,
    HNFTemplate * pHFingers
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>pHFingers</i>	[out] Pointer to a HNFTemplate that receives handle to created NTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHFingers</i> is NULL .
N_E_INVALID_OPERATION	NTemplate already contains fingers template. See NTemplateGetFingers function.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [HNFTemplate](#) | [NTemplateGetFingers](#)

6.5.1.5. NTemplateAddFingersCopy Function

Adds a copy of the fingers template to the NTemplate.

```
NResult N_API NTemplateAddFingersCopy(
    HNTemplate hTemplate,
    HNFTemplate hSrcFingers,
    HNFTemplate * pHFingers
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>hSrcFingers</i>	[in] Handle to the NTemplate object.
<i>pHFingers</i>	[out] Pointer to a HNFTemplate that receives handle to created NTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>hSrcFingers</i> or <i>pHFingers</i> is NULL .
N_E_INVALID_OPERATION	NTemplate already contains fingers template. See NTemplateGetFingers function.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [HNFTemplate](#) | [NTemplateGetFingers](#)

6.5.1.6. NTemplateAddFingersFromMemory Function

Unpacks a fingers template from the specified memory buffer and adds it to the NTemplate.

```
NResult N_API NTemplateAddFingersFromMemory(
    HNTemplate hTemplate,
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    HNFTemplate * pHFingers
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>buffer</i>	[in] Pointer to memory buffer that contains packed NTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NTemplate.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pHFingers</i>	[out] Pointer to HNFTemplate that receives handle to created NTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHFingers</i> , or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NTemplate format.

Error Code	Condition
N_E_INVALID_OPERATION	NTemplate already contains fingers template. See NTemplateGetFingers function.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see [NFTemplateCreateFromMemory](#) function.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [HNFTemplate](#) | [NTemplateGetFingers](#) | [NFTemplateCreateFromMemory](#)

6.5.1.7. NTemplateCalculateSize Function

Calculates the size of a packed NTemplate containing fingers and faces templates of the specified size.

```
NResult N_API NTemplateCalculateSize(
    NSizeType fingersTemplateSize,
    NSizeType facesTemplateSize,
    NSizeType * pSize
);
```

Parameters

<i>fingersTemplateSize</i>	[in] The size of packed fingers template.
<i>facesTemplateSize</i>	[in] The size of packed faces template.
<i>pSize</i>	[out] Pointer to NSizeType that receives calculated size of packed NTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>fingersTemplateSize</i> is not zero and is less than minimal size of packed fingers template or <i>facesTemplateSize</i> is not zero and is less than minimal size of packed faces template.
N_E_ARGUMENT_NULL	<i>pSize</i> is NULL .

Remarks

This is a low-level function and can be changed in future version of the library.

Use [NFTemplateCalculateSize](#) function to calculate packed size of a fingers template. Use function to calculate packed size of a faces template.

See Also

[NTemplate Module](#) | [NTemplateCalculateSize](#) | [NTemplateSaveToMemory](#) | [NTemplatePack](#)

6.5.1.8. NTemplateCheck Function

Checks if format of the packed NTemplate is correct.

```
NResult N_API NTemplateCheck(
    const void * buffer,
    NSizeType bufferSize
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory <i>buffer</i> buffer points to is inconsistent with NTemplate format.

See Also

[NTemplate Module](#)

6.5.1.9. NTemplateClear Function

Removes all templates from the NTemplate.

```
NResult N_API NTemplateClear(
    HNTemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
------------------	--------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

See Also

[NTemplate Module](#) | [HNTemplate](#)

6.5.1.10. NTemplateClone Function

Creates a copy of the NTemplate.

```
NResult N_API NTemplateClone(
    HNTemplate hTemplate,
    HNTemplate * pHClonedTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>pHClonedTemplate</i>	[out] Pointer to HNTemplate that receives handle to created NTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHClonedTemplate</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [NTemplateFree](#)

6.5.1.11. NTemplateCreate Function

Creates an empty NTemplate.

```
NResult N_API NTemplateCreate(
    HNTemplate * pHTemplate
);
```

Parameters

<i>pHTemplate</i>	[out] Pointer to HNTemplate that receives handle to created NTemplate object.
-------------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHTemplate</i> is NULL .

Error Code	Condition
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NTemplateFree](#) function.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [NTemplateFree](#)

6.5.1.12. NTemplateCreateFromMemory Function

Unpacks a NTemplate from the specified memory buffer.

```
NResult N_API NTemplateCreateFromMemory(
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NTemplateInfo * pInfo,
    HNTemplate * pHTemplate
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NTemplate.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pInfo</i>	[out] For internal use. Must be NULL .
<i>pHTemplate</i>	[out] Pointer to HNTemplate that receives handle to newly created NTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHTemplate</i> or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NTemplate format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NTemplateFree](#) function.

The following flags are supported:

- Flags supported by [NFTemplateCreateFromMemory](#) function are applied to fingers template (if) contained in the NTemplate.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [NTemplateInfo](#) | [NTemplateFree](#) | [NFTemplateCreateFromMemory](#) | [NTemplateSaveToMemory](#)

6.5.1.13. NTemplateFree Function

Deletes the NTemplate. After the object is deleted the specified handle is no longer valid.

```
void N_API NTemplateFree(
    HNTemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
------------------	--------------------------------------

Remarks

If *hTemplate* is [NULL](#), does nothing.

See Also

[NTemplate Module](#) | [HNTemplate](#)

6.5.1.14. NTemplateGetFaces Function

Retrieves the faces template of the NTemplate.

```
NResult N_API NTemplateGetFaces(
    HNTemplate hTemplate,
    * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>pValue</i>	[out] Points to that receives handle to NLTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .

Remarks

If the NTemplate does not contain face template, the returned handle is [NULL](#).

See Also

[NTemplate Module](#) | [HNTemplate](#)

6.5.1.15. NTemplateGetFingers Function

Retrieves the fingers template of the NTemplate.

```
NResult N_API NTemplateGetFingers(
    HNTemplate hTemplate,
    HNFTemplate * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>pValue</i>	[out] Points to HNFTemplate that receives handle to NTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .

Remarks

If the NTemplate does not contain fingers template, the returned handle is [NULL](#).

See Also

[NTemplate Module](#) | [HNTemplate](#)

6.5.1.16. NTemplateGetSize Function

Calculates packed size of the NTemplate.

```
NResult N_API NTemplateGetSize(
    HNTemplate hTemplate,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives calculated size of packed NTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pSize</i> is NULL .

Remarks

For the list of flags that are supported see [NTemplateSaveToMemory](#) function.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [NTemplateSaveToMemory](#)

6.5.1.17. NTemplatePack Function

Packs packed fingers and faces templates as NTemplate into the specified memory buffer.

```
NResult N_API NTemplatePack(
    const void * fingersTemplate,
    NSizeType fingersTemplateSize,
    const void * facesTemplate,
    NSizeType facesTemplateSize,
    void * buffer,
    NSizeType bufferSize,
    NSizeType * pSize
);
```

Parameters

<i>fingersTemplate</i>	[in] Pointer to memory buffer that contains packed fingers template.
<i>fingersTemplateSize</i>	[in] Size of memory buffer that contains packed fingers template.
<i>facesTemplate</i>	[in] Pointer to memory buffer that contains packed faces template.
<i>facesTemplateSize</i>	[in] Size of memory buffer that contains packed faces template.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NTemplate.
<i>bufferSize</i>	[in] Size of memory buffer to store packed NTemplate.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<p><i>bufferSize</i> is less than size required to store packed NTemplate.</p> <p>- or -</p> <p><i>fingersTemplateSize</i> is less than minimal fingers template size.</p> <p>- or -</p> <p><i>fingersTemplateSize</i> is not equal to size stored in memory buffer <i>fingersTemplate</i> points to.</p> <p>- or -</p> <p><i>facesTemplateSize</i> is less than minimal faces template size.</p> <p>- or -</p> <p><i>facesTemplateSize</i> is not equal to size stored in memory buffer <i>facesTemplate</i> points to.</p>
N_E_ARGUMENT_NULL	<p><i>fingersTemplate</i> is NULL and <i>fingersTemplateSize</i> is not equal to zero.</p> <p>- or -</p> <p><i>facesTemplate</i> is NULL and <i>facesTemplateSize</i> is not equal to zero.</p> <p>- or -</p> <p><i>buffer</i> or <i>pSize</i> is NULL.</p>
N_E_FORMAT	Data in memory buffer <i>fingersTemplate</i> points to is inconsistent with NTemplate format.

Remarks

bufferSize must not be less than value calculated with [NTemplateCalculateSize](#) function with the same parameter values.

See Also

[NTemplate Module](#) | [NTemplateCalculateSize](#) | [NTemplateUnpack](#)

6.5.1.18. NTemplateRemoveFaces Function

Removes faces template from the NTemplate.

```
NResult N_API NTemplateRemoveFaces(
    HNTemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
------------------	--------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

Remarks

If the NTemplate does not contain faces template, does nothing.

See Also

[HNTemplate](#)

6.5.1.19. NTemplateRemoveFingers Function

Removes fingers template from the NTemplate.

```
NResult N_API NTemplateRemoveFingers(
    HNTemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
------------------	--------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

Remarks

If the NTemplate does not contain fingers template, does nothing.

See Also

[NTemplate Module](#) | [HNTemplate](#)

6.5.1.20. NTemplateSaveToMemory Function

Packs the NTemplate into the specified memory buffer.

```
NResult N_API NTemplateSaveToMemory(
    HNTemplate hTemplate,
    void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NTemplate. Can be NULL .
<i>bufferSize</i>	[in] Size of memory buffer to store packed NTemplate.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>buffer</i> is not NULL and <i>bufferSize</i> is less than size required to store packed NTemplate.
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pSize</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is [NULL](#) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as [NTemplateGetSize](#) function.

If *buffer* is not [NULL](#), *bufferSize* must not be less than value calculated with [NTemplateGetSize](#) function.

The following flags are supported:

- Flags supported by [NFTemplateSaveToMemory](#) function are applied to fingers template (if) contained in the NTemplate.

See Also

[NTemplate Module](#) | [HNTemplate](#) | [NTemplateGetSize](#) | [NFTemplateSaveToMemory](#) | [NTemplateCreateFromMemory](#)

6.5.1.21. NTemplateUnpack Function

Unpacks packed fingers template from the packed NTemplate.

```
NResult N_API NTemplateUnpack(
    const void * buffer,
    NSizeType bufferSize,
    NByte * pMajorVersion,
    NByte * pMinorVersion,
    NUInt * pSize,
    NByte * pHeaderSize,
    const void * * pFingersTemplate,
    NSizeType * pFingersTemplateSize,
```

```
const void * * pFacesTemplate,
NSizeType * pFacesTemplateSize
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NTemplate.
<i>pMajorVersion</i>	[out] Pointer to NByte that receives major version of the packed NTemplate. Can be NULL .
<i>pMinorVersion</i>	[out] Pointer to NByte that receives minor version of the packed NTemplate. Can be NULL .
<i>pSize</i>	[out] Pointer to NUInt that receives size of the packed NTemplate. Can be NULL .
<i>pHeaderSize</i>	[out] Pointer to NByte that receives header size of the packed NTemplate. Can be NULL .
<i>pFingersTemplate</i>	[out] Pointer to void * that receives pointer to packed fingers template contained in the packed NTemplate. Can be NULL .
<i>pFingersTemplateSize</i>	[out] Pointer to NSizeType that receives size of packed fingers template contained in the packed NTemplate. Can be NULL .
<i>pFacesTemplate</i>	[out] Pointer to void * that receives pointer to packed faces template contained in the packed NTemplate. Can be NULL .
<i>pFacesTemplateSize</i>	[out] Pointer to NSizeType that receives size of packed faces template contained in the packed NTemplate. Can be NULL .

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory <i>buffer</i> buffer points to is inconsistent with NTemplate format.

See Also

[NTemplate Module](#) | [NTemplatePack](#)

6.5.2. NFRecord Module

Provides functionality for packing, unpacking and editing Neurotechnology Finger Records (NFRecords).

Header file: `NFRecord.h` and `NFRecordV1.h`

Functions

<code>NFRecordAddCore</code>	Adds a NFCore to the end of <code>NFRecord</code> cores.
<code>NFRecordAddDelta</code>	Adds a NFDelta to the end of <code>NFRecord</code> deltas.
<code>NFRecordAddDoubleCore</code>	Adds a NFDoubleCore to the end of <code>NFRecord</code> double cores.
<code>NFRecordAddMinutia</code>	Adds a NFMinutia to the end of <code>NFRecord</code> minutiae.
<code>NFRecordCheck</code>	Checks if format of the packed <code>NFRecord</code> is correct.
<code>NFRecordClearCores</code>	Removes all cores from the <code>NFRecord</code> .
<code>NFRecordClearDeltas</code>	Removes all deltas from the <code>NFRecord</code> .
<code>NFRecordClearDoubleCores</code>	Removes all double cores from the <code>NFRecord</code> .
<code>NFRecordClearMinutiae</code>	Removes all minutiae from the <code>NFRecord</code> .
<code>NFRecordClone</code>	Creates a copy of the <code>NFRecord</code> .
<code>NFRecordCreate</code>	Creates an empty <code>NFRecord</code> .
<code>NFRecordCreateFromMemory</code>	Unpacks a <code>NFRecord</code> from the specified memory buffer.
<code>NFRecordFree</code>	Deletes the <code>NFRecord</code> . After the object is deleted the specified handle is no longer valid.
<code>NFRecordGetCbeffProductType</code>	Retrieves the Cbeff product type of the <code>NFRecord</code> .
<code>NFRecordGetCbeffProductTypeMem</code>	Retrieves the Cbeff product type of the packed <code>NFRecord</code> .
<code>NFRecordGetCore</code>	Retrieves the core at the specified index of the <code>NFRecord</code> .
<code>NFRecordGetCoreCapacity</code>	Retrieves the number of cores that the <code>NFRecord</code> can contain.
<code>NFRecordGetCoreCount</code>	Retrieves the number of cores in the <code>NFRecord</code> .
<code>NFRecordGetCores</code>	Copies all cores of <code>NFRecord</code> to the specified array.
<code>NFRecordGetDelta</code>	Retrieves the delta at the specified index of the <code>NFRecord</code> .
<code>NFRecordGetDeltaCapacity</code>	Retrieves the number of deltas that the <code>NFRecord</code> can contain.
<code>NFRecordGetDeltaCount</code>	Retrieves the number of deltas in the <code>NFRecord</code> .
<code>NFRecordGetDeltas</code>	Copies all deltas of <code>NFRecord</code> to the specified array.
<code>NFRecordGetDoubleCore</code>	Retrieves the double core at the specified index of the <code>NFRecord</code> .
<code>NFRecordGetDoubleCoreCapacity</code>	Retrieves the number of double cores that the <code>NFRecord</code> can contain.
<code>NFRecordGetDoubleCoreCount</code>	Retrieves the number of double cores in the <code>NFRecord</code> .
<code>NFRecordGetDoubleCores</code>	Copies all double cores of <code>NFRecord</code> to the specified array.

NFRecordGetG	Retrieves the G of the NFRecord.
NFRecordGetGMem	Retrieves the G of the packed NFRecord.
NFRecordGetHeight	Retrieves the height of the image the NFRecord is made from.
NFRecordGetHeightMem	Retrieves the height of the image the packed NFRecord is made from.
NFRecordGetHorzResolution	Retrieves the horizontal resolution in dpi of the image the NFRecord is made from.
NFRecordGetHorzResolutionMem	Retrieves the horizontal resolution of the image the packed NFRecord is made from.
NFRecordGetImpressionType	Retrieves the impression type of the NFRecord.
NFRecordGetImpressionTypeMem	Retrieves the impression type of the packed NFRecord.
NFRecordGetMaxSize	Retrieves the maximal size of a packed NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.
NFRecordGetMaxSizeV1	Retrieves the maximal size of a packed in version 1.0 format NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.
NFRecordGetMinutia	Retrieves the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaCapacity	Retrieves the number of minutiae that the NFRecord can contain.
NFRecordGetMinutiaCount	Retrieves the number of minutiae in the NFRecord.
NFRecordGetMinutiaFormat	Retrieves the format of the minutiae in NFRecord.
NFRecordGetMinutiaNeighbor	Retrieves the minutia neighbor at the specified index of the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaNeighborCount	Retrieves the number of minutia neighbors in the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaNeighbors	Copies all minutia neighbors of the minutia at the specified index of the NFRecord to the specified array.
NFRecordGetMinutiae	Copies all minutiae of NFRecord to the specified array.
NFRecordGetPatternClass	Retrieves the pattern class of the NFRecord.
NFRecordGetPatternClassMem	Retrieves the pattern class of the packed NFRecord.
NFRecordGetPosition	Retrieves the finger position of the NFRecord.
NFRecordGetPositionMem	Retrieves the finger position of the packed NFRecord.
NFRecordGetQuality	Retrieves the quality of the NFRecord.
NFRecordGetQualityMem	Retrieves the quality of the packed NFRecord.
NFRecordGetRidgeCountsType	Retrieves the ridge counts type the NFRecord contains.
NFRecordGetSize	Calculates packed size of the NFRecord.
NFRecordGetSizeV1	Calculates packed in version 1.0 format size of the NFRecord.

NFRecordGetVertResolution	Retrieves the vertical resolution of the image the NFRecord is made from.
NFRecordGetVertResolutionMem	Retrieves the vertical resolution of the image the packed NFRecord is made from.
NFRecordGetWidth	Retrieves the width of the image the NFRecord is made from.
NFRecordGetWidthMem	Retrieves the width of the image the packed NFRecord is made from.
NFRecordInfoDispose	For internal use.
NFRecordInsertCore	Inserts a NFCore into the NFRecord at the specified index.
NFRecordInsertDelta	Inserts a NFDelta into the NFRecord at the specified index.
NFRecordInsertDoubleCore	Inserts a NFDoubleCore into the NFRecord at the specified index.
NFRecordInsertMinutia	Inserts a NFMinutia into the NFRecord at the specified index.
NFRecordRemoveCore	Removes the core at the specified index of the NFRecord.
NFRecordRemoveDelta	Removes the delta at the specified index of the NFRecord.
NFRecordRemoveDoubleCore	Removes the double core at the specified index of the NFRecord.
NFRecordRemoveMinutia	Removes the minutia at the specified index of the NFRecord.
NFRecordSaveToMemory	Packs the NFRecord into the specified memory buffer.
NFRecordSaveToMemoryV1	Packs the NFRecord into the specified memory buffer in version 1.0 format.
NFRecordSetCbeffProductType	Sets the Cbeff product type.
NFRecordSetCore	Sets a NFCore at the specified index of the NFRecord.
NFRecordSetCoreCapacity	Sets the number of cores that the NFRecord can contain.
NFRecordSetDelta	Sets a NFDelta at the specified index of the NFRecord.
NFRecordSetDeltaCapacity	Sets the number of deltas that the NFRecord can contain.
NFRecordSetDoubleCore	Sets a NFDoubleCore at the specified index of the NFRecord.
NFRecordSetDoubleCoreCapacity	Sets the number of double cores that the NFRecord can contain.
NFRecordSetG	Sets the G of the NFRecord.
NFRecordSetImpressionType	Sets the impression type of the NFRecord.
NFRecordSetMinutia	Sets a NFMinutia at the specified index of the NFRecord.
NFRecordSetMinutiaCapacity	Sets the number of minutiae that the NFRecord can contain.

NFRecordSetMinutiaNeighbor	Sets a NFMinutiaNeighbor at the specified index of the minutia at the specified index of the NFRecord.
NFRecordSetMinutiaFormat	Sets the format of the minutiae in NFRecord.
NFRecordSetPatternClass	Sets the pattern class of the NFRecord.
NFRecordSetPosition	Sets the finger position of the NFRecord.
NFRecordSetQuality	Sets the quality of the NFRecord.
NFRecordSetRidgeCountsType	Sets the ridge counts type the NFRecord contains.
NFRecordSortMinutiae	Sorts minutiae in NFRecord by the specified order.
NFRecordTruncateMinutiae	Truncates minutiae in NFRecord by peeling off the minutiae convex hull while minutia count is greater than the specified maximal count.
NFRecordTruncateMinutiaeByQuality	Truncates minutiae in NFRecord by removing minutiae which NFMinutia.Quality field value is less than the specified threshold while minutia count is greater than the specified maximal count.

Structures

NFCore	Represents a core in a NFRecord.
NFDelta	Represents a delta in a NFRecord.
NFDoubleCore	Represents a double core in a NFRecord.
NFMinutia	Represents a minutia in a NFRecord.
NFMinutiaNeighbor	Represents a minutia neighbor in a NFRecord.
NFRecordInfo	For internal use.

Enumerations

NFImpressionType	Specifies the impression type.
NFMinutiaFormat	Specifies the minutia format.
NFMinutiaOrder	Specifies minutia order.
NFMinutiaType	Specifies the minutia type.
NFPatternClass	Specifies the pattern class of the fingerprint.
NFPosition	Specifies the finger position.
NFRidgeCountsType	Specifies the type of ridge counts contained in a NFRecord.

Types

HNFRecord	Handle to NFRecord object.
---------------------------	----------------------------

Macros

NFR_BLOCK_SIZE	For internal use.
NFR_MAX_BLOCKED_ORIENTS_DIMENSION	For internal use.
NFR_MAX_CORE_COUNT	The maximum number of cores a NFRecord can contain.
NFR_MAX_DELTA_COUNT	The maximum number of deltas a NFRecord can contain.
NFR_MAX_DIMENSION	The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.
NFR_MAX_DOUBLE_CORE_COUNT	The maximum number of double cores a NFRecord can contain.
NFR_MAX_MINUTIA_COUNT	The maximum number of minutiae a NFRecord can contain.
NFR_RESOLUTION	The resolution of minutiae in dpi, cores, deltas and double cores coordinates in a NFRecord.
NFR_SAVE_BLOCKED_ORIENTS	The flag indicating whether blocked orientations should be packed in NFRecord.
NFR_SKIP_BLOCKED_ORIENTS	The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.
NFR_SKIP_CURVATURES	The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.
NFR_SKIP_GS	The flag indicating whether minutiae gs should be skipped while unpacking or packing NFRecord.
NFR_SKIP_QUALITIES	The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.
NFR_SKIP_RIDGE_COUNTS	The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.
NFR_SKIP_SINGULAR_POINTS	The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.

See Also

[NFRecord Module](#)

6.5.2.1. NFCore Structure

Represents a core in a NFRecord.

```
typedef struct NFCore_ { } NFCore;
```

Fields

<i>Angle</i>	Angle of this NFCore .
<i>X</i>	X coordinate of this NFCore .
<i>Y</i>	Y coordinate of this NFCore .

See Also

[NFRecord Module](#)

6.5.2.1.1. NFCore.Angle Field

Angle of this [NFCore](#).

```
NInt Angle;
```

Remarks

The angle of the core is specified in 180/128 degrees units in counterclockwise order and cannot be less than zero or greater than 256 minus one.

The value of -1 can be specified if the angle of the core is unknown.

See Also

[NFCore](#)

6.5.2.1.2. NFCore.X Field

X coordinate of this [NFCore](#).

```
NUShort X;
```

Remarks

The x coordinate of the core is specified in pixels at [NFR_RESOLUTION](#) and $X * [\text{NFRecord horizontal resolution}] / \text{NFR_RESOLUTION}$ cannot be greater than [NFR_MAX_DIMENSION](#) or NFRecord width minus one.

See Also

[NFCore](#)

6.5.2.1.3. NFCore.Y Field

Y coordinate of this [NFCore](#).

```
NUShort Y;
```

Remarks

The y coordinate of the core is specified in pixels at [NFR_RESOLUTION](#) and $Y * [\text{NFRecord vertical resolution}] / \text{NFR_RESOLUTION}$ cannot be greater than [NFR_MAX_DIMENSION](#) or NFRecord height minus one.

See Also

[NFCore](#)

6.5.2.2. NFDelta Structure

Represents a delta in a NFRecord.

```
typedef struct NFDelta_ { } NFDelta;
```

Fields

Angle1	First angle of this NFDelta .
Angle2	Second angle of this NFDelta .

Angle3	Third angle of this NFDelta .
X	X coordinate of this NFDelta .
Y	Y coordinate of this NFDelta .

See Also[NFRecord Module](#)**6.5.2.2.1. NFDelta.Angle1 Field**First angle of this [NFDelta](#).

```
NInt Angle1;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and cannot be less than zero or greater than 256 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

See Also[NFDelta](#)**6.5.2.2.2. NFDelta.Angle2 Field**Second angle of this [NFDelta](#).

```
NInt Angle2;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and cannot be less than zero or greater than 256 minus one.

The value of -1 can be specified if the second angle of the delta is unknown.

See Also[NFDelta](#)**6.5.2.2.3. NFDelta.Angle3 Field**Third angle of this [NFDelta](#).

```
NInt Angle3;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and cannot be less than zero or greater than 256 minus one.

The value of -1 can be specified if the third angle of the delta is unknown.

See Also[NFDelta](#)**6.5.2.2.4. NFDelta.X Field**

X coordinate of this [NFDelta](#).

```
NUShort X;
```

Remarks

The x coordinate of the delta is specified in pixels at [NFR_RESOLUTION](#) and $X * [\text{NFRRecord horizontal resolution}] / \text{NFR_RESOLUTION}$ cannot be greater than [NFR_MAX_DIMENSION](#) or NFRRecord width minus one.

See Also

[NFDelta](#)

6.5.2.2.5. NFDelta.Y Field

Y coordinate of this [NFDelta](#).

```
NUShort Y;
```

Remarks

The y coordinate of the delta is specified in pixels at [NFR_RESOLUTION](#) and $Y * [\text{NFRRecord vertical resolution}] / \text{NFR_RESOLUTION}$ cannot be greater than [NFR_MAX_DIMENSION](#) or NFRRecord height minus one.

See Also

[NFDelta](#)

6.5.2.3. NFDoubleCore Structure

Represents a double core in a NFRRecord.

```
typedef struct NFDoubleCore_ { } NFDoubleCore;
```

Fields

X	X coordinate of this NFDoubleCore .
Y	Y coordinate of this NFDoubleCore .

See Also

[NFRRecord Module](#)

6.5.2.3.1. NFDoubleCore.X Field

X coordinate of this [NFDoubleCore](#).

```
NUShort X;
```

Remarks

The x coordinate of the double core is specified in pixels at [NFR_RESOLUTION](#) and $X * [\text{NFRRecord horizontal resolution}] / \text{NFR_RESOLUTION}$ cannot be greater than [NFR_MAX_DIMENSION](#) or NFRRecord width minus one.

See Also

[NFDoubleCore](#)

6.5.2.3.2. NFDoubleCore.Y Field

Y coordinate of this [NFDoubleCore](#).

```
NUShort Y;
```

Remarks

The y coordinate of the double core is specified in pixels at [NFR_RESOLUTION](#) and $Y * [\text{NFRecord vertical resolution}] / \text{NFR_RESOLUTION}$ cannot be greater than [NFR_MAX_DIMENSION](#) or NFRecord height minus one.

See Also

[NFDoubleCore](#)

6.5.2.4. NFImpressionType Enumeration

Specifies the impression type.

```
typedef enum NFImpressionType_ { } NFImpressionType;
```

Members

nfitLatentImpression	Latent impression fingerprint.
nfitLatentLift	Latent lift fingerprint.
nfitLatentPhoto	Latent photo fingerprint.
nfitLatentTracing	Latent tracing fingerprint.
nfitLiveScanContactless	Live-scanned fingerprint using contactless device.
nfitLiveScanPlain	Live-scanned plain fingerprint.
nfitLiveScanRolled	Live-scanned rolled fingerprint.
nfitNonliveScanPlain	Nonlive-scanned (from paper) plain fingerprint.
nfitNonliveScanRolled	Nonlive-scanned (from paper) rolled fingerprint.
nfitSwipe	Live-scanned fingerprint by sliding the finger across a "swipe" sensor.

Remarks

This enumeration is implemented according to ANSI/NIST-ITL 1-2000 and ANSI INCITS 378-2004 standards.

See Also

[NFRecord Module](#)

6.5.2.5. NFMinutia Structure

Represents a minutia in a NFRecord.

```
typedef struct NFMinutia_ { } NFMinutia;
```

Fields

Angle	Angle of this NFMinutia .
-----------------------	---

<i>Curvature</i>	Ridge curvature near this NFMinutia .
<i>G</i>	G (ridge density) near this NFMinutia .
<i>Quality</i>	Quality of this NFMinutia .
<i>Type</i>	Type of this NFMinutia .
<i>X</i>	X coordinate of this NFMinutia .
<i>Y</i>	Y coordinate of this NFMinutia .

See Also

[NFRecord Module](#)

6.5.2.5.1. NFMinutia.Angle Field

Angle of this [NFMinutia](#).

```
NByte Angle;
```

Remarks

The angle of the minutia is specified in 180/128 degrees units in counterclockwise order and cannot be greater than 256 minus one.

See Also

[NFMinutia](#)

6.5.2.5.2. NFMinutia.Curvature Field

Ridge curvature near this [NFMinutia](#).

```
NByte Curvature;
```

Remarks

If curvature of the minutia is unknown it must be set to 255.

See Also

[NFMinutia](#)

6.5.2.5.3. NFMinutia.G Field

G (ridge density) near this [NFMinutia](#).

```
NByte G;
```

Remarks

If G of the minutia is unknown it must be set to 255.

See Also

[NFMinutia](#)

6.5.2.5.4. NFMinutia.Quality Field

Quality of this [NFMinutia](#).

```
NByte Quality;
```

Remarks

The quality of the minutia must be in the range [0, 100]. The higher it is, the better the quality of the minutia is.

If quality of the minutia is unknown it must be set to zero.

See Also

[NFMinutia](#)

6.5.2.5.5. NFMinutia.Type Field

Type of this [NFMinutia](#).

```
NFMinutiaType Type;
```

See Also

[NFMinutia](#)

6.5.2.5.6. NFMinutia.X Field

X coordinate of this [NFMinutia](#).

```
NUShort X;
```

Remarks

The x coordinate of the minutia is specified in pixels at [NFR_RESOLUTION](#) and $X * [\text{NFRecord horizontal resolution}] / \text{NFR_RESOLUTION}$ cannot be greater than [NFR_MAX_DIMENSION](#) or NFRecord width minus one.

See Also

[NFMinutia](#)

6.5.2.5.7. NFMinutia.Y Field

Y coordinate of this [NFMinutia](#).

```
NUShort Y;
```

Remarks

The y coordinate of the minutia is specified in pixels at [NFR_RESOLUTION](#) and $Y * [\text{NFRecord vertical resolution}] / \text{NFR_RESOLUTION}$ cannot be greater than [NFR_MAX_DIMENSION](#) or NFRecord height minus one.

See Also

[NFMinutia](#)

6.5.2.6. NFMinutiaFormat Enumeration

Specifies the minutia format.

This enumeration allows a bitwise combination of its member values.

```
typedef enum NFMinutiaFormat_ { } NFMinutiaFormat;
```

Members

<code>nfmfHasCurvature</code>	Indicates that NFMinutia . Curvature field contains meaningful value and is preserved during unpacking/packing of NFRecord.
<code>nfmfHasG</code>	Indicates that NFMinutia . G field contains meaningful value and is preserved during unpacking/packing of NFRecord.
<code>nfmfHasQuality</code>	Indicates that NFMinutia . Quality field contains meaningful value and is preserved during unpacking/packing of NFRecord.

See Also

[NFRecord Module](#) | [NFMinutia](#)

6.5.2.7. NFMinutiaNeighbor Structure

Represents a minutia neighbor in a NFRecord.

```
typedef struct NFMinutiaNeighbor_ { } NFMinutiaNeighbor;
```

Fields

Index	Index of neighbor minutia.
RidgeCount	Ridge count to neighbor minutia.

See Also

[NFRecord Module](#)

6.5.2.7.1. NFMinutiaNeighbor.Index Field

Index of neighbor minutia.

```
NInt Index;
```

See Also

[NFMinutiaNeighbor](#)

6.5.2.7.2. NFMinutiaNeighbor.RidgeCount Field

Ridge count to neighbor minutia.

```
NByte RidgeCount;
```

See Also

[NFMinutiaNeighbor](#)

6.5.2.8. NFMinutiaOrder Enumeration

Specifies minutia order.

```
typedef enum NFMinutiaOrder_ { } NFMinutiaOrder;
```

Members

nfmoAscending	Specifies than minutiae are sorted ascending by the specified order.
nfmoDescending	Specifies than minutiae are sorted descending by the specified order.
nfmoCartesianXY	Specifies than minutiae are sorted by NFMinutia.X field. If NFMinutia.X field of two minutiae are equal NFMinutia.Y field is compared.
nfmoCartesianYX	Specifies than minutiae are sorted by NFMinutia.Y field. If NFMinutia.Y field of two minutiae are equal NFMinutia.X is compared.
nfmoAngle	Specifies than minutiae are sorted by NFMinutia.Angle field.
nfmoPolar	Specifies than minutiae are sorted by distance from minutiae center of mass. If distance of two minutiae are equal NFMinutia.Angle field is compared.

See Also

[NFRecord Module](#)

6.5.2.9. NFMinutiaType Enumeration

Specifies the minutia type.

```
typedef enum NFMinutiaType_ { } NFMinutiaType;
```

Members

nfmtBifurcation	The minutia that is a bifurcation of a ridge.
nfmtEnd	The minutia that is an end of a ridge.
nfmtUnknown	The type of the minutia is unknown.

See Also

[NFRecord Module](#) | [NFMinutia](#)

6.5.2.10. NFPatternClass Enumeration

Specifies the pattern class of the fingerprint.

```
typedef enum NFPatternClass_ { } NFPatternClass;
```

Members

nfpcAccidentalWhorl	Accidental whorl pattern class.
nfpcAmputation	Amputation. Pattern class is not available.
nfpcCentralPocketLoop	Central pocket loop pattern class.
nfpcDoubleLoop	Double loop pattern class.

nfpcLeftSlantLoop	Left slant loop pattern class.
nfpcPlainArch	Plain arch pattern class.
nfpcPlainWhorl	Plain whorl pattern class.
nfpcRadialLoop	Radial loop pattern class.
nfpcRightSlantLoop	Right slant loop pattern class.
nfpcScar	Scar. Pattern class is not available.
nfpcTentedArch	Tented arch pattern class.
nfpcUlnarLoop	Ulnar loop pattern class.
nfpcUnknown	Unknown pattern class.
nfpcWhorl	Whorl pattern class.

Remarks

This enumeration is implemented according to ANSI/NIST-ITL 1-2000 standard.

See Also

[NFRecord Module](#)

6.5.2.11. NFPosition Enumeration

Specifies the finger position.

```
typedef enum NFPosition_ { } NFPosition;
```

Members

nfpLeftIndex	Index finger of the left hand.
nfpLeftLittle	Little finger of the left hand.
nfpLeftMiddle	Middle finger of the left hand.
nfpLeftRing	Ring finger of the left hand.
nfpLeftThumb	Thumb of the left hand.
nfpRightIndex	Index finger of the right hand.
nfpRightLittle	Little finger of the right hand.
nfpRightMiddle	Middle finger of the right hand.
nfpRightRing	Ring finger of the right hand.
nfpRightThumb	Thumb of the right hand.
nfpUnknown	Unknown finger.

Remarks

This enumeration is implemented according to ANSI/NIST-ITL 1-2000 and ANSI INCITS 378-2004 standards.

See Also

[NFRecord Module](#)

6.5.2.12. NFRecordAddCore Function

Adds a [NFCore](#) to the end of NFRecord cores.

```
NResult N_API NFRecordAddCore(
    HNFRecord hRecord,
    const NFCore * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[in] Pointer to the NFCore to add.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_INVALID_OPERATION	Number of cores in NFRecord (see NFRecordGet-CoreCount) is equal to NFR_MAX_CORE_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFCore](#)

6.5.2.13. NFRecordAddDelta Function

Adds a [NFDelta](#) to the end of NFRecord deltas.

```
NResult N_API NFRecordAddDelta(
    HNFRecord hRecord,
    const NFDelta * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[in] Pointer to the NFDelta to add.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_INVALID_OPERATION	Number of deltas in <i>NFRecord</i> (see NFRecordGetDeltaCount) is equal to NFR_MAX_DELTA_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFDelta](#)

6.5.2.14. NFRecordAddDoubleCore Function

Adds a [NFDoubleCore](#) to the end of *NFRecord* double cores.

```
NResult N_API NFRecordAddDoubleCore(
    HNFRecord hRecord,
    const NFDoubleCore * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the <i>NFRecord</i> object.
<i>pValue</i>	[in] Pointer to the NFDoubleCore to add.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_INVALID_OPERATION	Number of double cores in <i>NFRecord</i> (see NFRecordGetDoubleCoreCount) is equal to NFR_MAX_DOUBLE_CORE_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFDoubleCore](#)

6.5.2.15. NFRecordAddMinutia Function

Adds a [NFMinutia](#) to the end of *NFRecord* minutiae.

```
NResult N_API NFRecordAddMinutia(
    HNFRecord hRecord,
    const NFMinutia * pValue
);
```


Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[in] Pointer to the NFMinutia to add.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_INVALID_OPERATION	Number of minutiae in NFRecord (see NFRecordGetMinutiaCount) is equal to NFR_MAX_MINUTIA_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFMinutia](#)

6.5.2.16. NFRecordCheck Function

Checks if format of the packed NFRecord is correct.

```
NResult N_API NFRecordCheck(
    const void * buffer,
    NSizeType bufferSize
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

This function supports both NfRecord version 1.0 and 2.0 formats.

See Also

[NfRecord Module](#)

6.5.2.17. NfRecordClearCores Function

Removes all cores from the NfRecord.

```
NResult N_API NfRecordClearCores(
    HNFRecord hRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
----------------	-------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NfRecord Module](#) | [HNFRecord](#)

6.5.2.18. NfRecordClearDeltas Function

Removes all deltas from the NfRecord.

```
NResult N_API NfRecordClearDeltas(
    HNFRecord hRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
----------------	-------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#)

6.5.2.19. NFRecordClearDoubleCores Function

Removes all double cores from the NFRecord.

```
NResult N_API NFRecordClearDoubleCores(
    HNFRecord hRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
----------------	-------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#)

6.5.2.20. NFRecordClearMinutiae Function

Removes all minutiae from the NFRecord.

```
NResult N_API NFRecordClearMinutiae(
    HNFRecord hRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
----------------	-------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#)

6.5.2.21. NFRecordClone Function

Creates a copy of the NFRecord.

```
NResult N_API NFRecordClone(
    HNFRecord hRecord,
    HNFRecord * pHClonedRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pHClonedRecord</i>	[out] Pointer to a HNFRecord that receives handle to newly created NFRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pHClonedRecord</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NFRecordFree](#) function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordFree](#)

6.5.2.22. NFRecordCreate Function

Creates an empty NFRecord.

```
NResult N_API NFRecordCreate(
    NUShort width,
    NUShort height,
    NUShort horzResolution,
    NUShort vertResolution,
    NUInt flags,
    HNFRecord * pHRecord
);
```

Parameters

<i>width</i>	[in] Specifies width of fingerprint image.
<i>height</i>	[in] Specifies height of fingerprint image.
<i>horzResolution</i>	[in] Specifies horizontal resolution in pixels per inch of fingerprint image.
<i>vertResolution</i>	[in] Specifies vertical resolution in pixels per inch of fingerprint image.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function. This parameter is reserved, must be zero.

<i>pHRecord</i>	[out] Pointer to HNFRecord that receives handle to created NRecord object.
-----------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is zero.
N_E_ARGUMENT_NULL	<i>pHRecord</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NRecordFree](#) function.

See Also

[NRecord Module](#) | [HNFRecord](#) | [NRecordFree](#)

6.5.2.23. NRecordCreateFromMemory Function

Unpacks a NRecord from the specified memory buffer.

```
NResult N_API NRecordCreateFromMemory(
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NRecordInfo * pInfo,
    HNFRecord * pHRecord
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NRecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pInfo</i>	[out] For internal use. Must be NULL .
<i>pHRecord</i>	[out] Pointer to HNFRecord that receives handle to newly created NRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHRecord</i> or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NRecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- [NFR_SKIP_BLOCKED_ORIENTS](#)
- [NFR_SKIP_CURVATURES](#)
- [NFR_SKIP_GS](#)
- [NFR_SKIP_QUALITIES](#)
- [NFR_SKIP_RIDGE_COUNTS](#)
- [NFR_SKIP_SINGULAR_POINTS](#)

This function supports both NRecord version 1.0 and 2.0 formats.

Created object must be deleted using [NRecordFree](#) function.

See Also

[NRecord Module](#) | [HNRecord](#) | [NRecordInfo](#) | [NRecordFree](#) | [NRecordSaveToMemory](#)

6.5.2.24. NRecordFree Function

Deletes the NRecord. After the object is deleted the specified handle is no longer valid.

```
void N_API NRecordFree(
    HNRecord hRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
----------------	------------------------------------

Remarks

If *hRecord* is [NULL](#), does nothing.

See Also

[NRecord Module](#) | [HNRecord](#)

6.5.2.25. NRecordGetCbeffProductType Function

Retrieves the Cbeff product type of the NRecord.

```
NResult N_API NRecordGetCbeffProductType(
    HNRecord hRecord,
    NUShort * pValue
```

```
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
<i>pValue</i>	[out] Pointer to NUShort that receives Cbeff product type.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NRecord Module](#) | [HNRecord](#) | [NRecordSetCbeffProductType](#) [NRecordGetCbeffProductTypeMem](#)

6.5.2.26. NRecordGetCbeffProductTypeMem Function

Retrieves the Cbeff product type of the packed NRecord.

```
NResult N_API NRecordGetCbeffProductTypeMem(
    const void * buffer,
    NSizeType bufferSize,
    NUShort * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NRecord.
<i>pValue</i>	[out] Pointer to NUShort that receives Cbeff product type.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NRecord Module](#) | [HNRecord](#) | [NRecordSetCbeffProductType](#) [NRecordGetCbeffProductType](#)

6.5.2.27. NFRecordGetCore Function

Retrieves the core at the specified index of the NFRecord.

```
NResult N_API NFRecordGetCore(
    HNFRecord hRecord,
    NInt index,
    NFCore * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of core to retrieve.
<i>pValue</i>	[out] Pointer to NFCore that receives core.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFCore](#) | [NFRecordGetCoreCount](#) | [NFRecordSetCore](#)

6.5.2.28. NFRecordGetCoreCapacity Function

Retrieves the number of cores that the NFRecord can contain.

```
NResult N_API NFRecordGetCoreCapacity(
    HNFRecord hRecord,
    NInt * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NInt that receives number of cores NFRecord can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

Core capacity is the number of cores that the `NFRecord` can store. Core count (see [NFRecordGetCoreCount](#) function) is the number of cores that are actually in the `NFRecord`.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordSetCoreCapacity](#) | [NFRecordGetCoreCount](#)

6.5.2.29. NFRecordGetCoreCount Function

Retrieves the number of cores in the `NFRecord`.

```
NResult N_API NFRecordGetCoreCount(
    HNFRecord hRecord,
    NInt * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the <code>NFRecord</code> object.
<i>pValue</i>	[out] Pointer to NInt that receives number of cores.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

Core capacity (see [NFRecordGetCoreCapacity](#) and [NFRecordSetCoreCapacity](#) functions) is the number of cores that the `NFRecord` can store. Core count is the number of cores that are actually in the `NFRecord`.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetCoreCapacity](#) | [NFRecordSetCoreCapacity](#)

6.5.2.30. NFRecordGetCores Function

Copies all cores of `NFRecord` to the specified array.

```
NResult N_API NFRecordGetCores(
    HNFRecord hRecord,
    NFCore * arCores
);
```

```
);
```

Parameters

<i>hRecord</i>	[in] Handle to the <code>NFRecord</code> object.
<i>arCores</i>	[out] Pointer to array of <code>NFCore</code> that receives cores.

Return Values

If the function succeeds, the return value is `N_OK`.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
<code>N_E_ARGUMENT_NULL</code>	<i>hRecord</i> or <i>arCores</i> is <code>NULL</code> .

Remarks

Array *arCores* points to must be large enough to receive all `NFRecord` cores. See `NFRecordGetCoreCount` function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFCore](#) | [NFRecordGetCoreCount](#)

6.5.2.31. NFRecordGetDelta Function

Retrieves the delta at the specified index of the `NFRecord`.

```
NResult N_API NFRecordGetDelta(
    HNFRecord hRecord,
    NInt index,
    NFDelta * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the <code>NFRecord</code> object.
<i>index</i>	[in] Index of delta to retrieve.
<i>pValue</i>	[out] Pointer to <code>NFDelta</code> that receives delta.

Return Values

If the function succeeds, the return value is `N_OK`.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
<code>N_E_ARGUMENT_NULL</code>	<i>hRecord</i> or <i>pValue</i> is <code>NULL</code> .
<code>N_E_ARGUMENT_OUT_OF_RANGE</code>	<i>index</i> is less than zero or greater than or equal to delta count obtained using <code>NFRecordGetDeltaCount</code> function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFDelta](#) | [NFRecordGetDeltaCount](#) | [NFRecordSetDelta](#)

6.5.2.32. NFRecordGetDeltaCapacity Function

Retrieves the number of deltas that the NFRecord can contain.

```
NResult N_API NFRecordGetDeltaCapacity(
    HNFRecord hRecord,
    NInt * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NInt that receives number of deltas NFRecord can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

Delta capacity is the number of deltas that the NFRecord can store. Delta count (see [NFRecordGetDeltaCount](#) function) is the number of deltas that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordSetDeltaCapacity](#) | [NFRecordGetDeltaCount](#)

6.5.2.33. NFRecordGetDeltaCount Function

Retrieves the number of deltas in the NFRecord.

```
NResult N_API NFRecordGetDeltaCount(
    HNFRecord hRecord,
    NInt * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NInt that receives number of deltas.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

Delta capacity (see [NFRecordGetDeltaCapacity](#) and [NFRecordSetDeltaCapacity](#) functions) is the number of deltas that the NFRecord can store. Delta count is the number of deltas that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetDeltaCapacity](#) | [NFRecordSetDeltaCapacity](#)

6.5.2.34. NFRecordGetDeltas Function

Copies all deltas of NFRecord to the specified array.

```
NResult N_API NFRecordGetDeltas(
    HNFRecord hRecord,
    NFDelta * arDeltas
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>arDeltas</i>	[out] Pointer to array of NFDelta that receives deltas.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>arDeltas</i> is NULL .

Remarks

Array *arDeltas* points to must be large enough to receive all NFRecord deltas. See [NFRecordGetDeltaCount](#) function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFDelta](#) | [NFRecordGetDeltaCount](#)

6.5.2.35. NFRecordGetDoubleCore Function

Retrieves the double core at the specified index of the NFRecord.

```
NResult N_API NFRecordGetDoubleCore(
    HNFRecord hRecord,
```

```

    NInt index,
    NFDoubleCore * pValue
);

```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
<i>index</i>	[in] Index of double core to retrieve.
<i>pValue</i>	[out] Pointer to NFDoubleCore that receives double core.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to double core count obtained using NRecordGetDoubleCoreCount function.

See Also

[NRecord Module](#) | [HNRecord](#) | [NFDoubleCore](#) | [NRecordGetDoubleCoreCount](#) | [NRecordSetDoubleCore](#)

6.5.2.36. NRecordGetDoubleCoreCapacity Function

Retrieves the number of double cores that the NRecord can contain.

```

NResult N_API NRecordGetDoubleCoreCapacity(
    HNRecord hRecord,
    NInt * pValue
);

```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
<i>pValue</i>	[out] Pointer to NInt that receives number of double cores NRecord can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

Double core capacity is the number of double cores that the NFRecord can store. Double core count (see [NFRecordGetDoubleCoreCount](#) function) is the number of double cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding double cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordSetDoubleCoreCapacity](#) | [NFRecordGetDoubleCoreCount](#)

6.5.2.37. NFRecordGetDoubleCoreCount Function

Retrieves the number of double cores in the NFRecord.

```
NResult N_API NFRecordGetDoubleCoreCount(
    HNFRecord hRecord,
    NInt * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NInt that receives number of double cores.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

Double core capacity (see [NFRecordGetDoubleCoreCapacity](#) and [NFRecordSetDoubleCoreCapacity](#) functions) is the number of double cores that the NFRecord can store. Double core count is the number of double cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding double cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetDoubleCoreCapacity](#) | [NFRecordSetDoubleCoreCapacity](#)

6.5.2.38. NFRecordGetDoubleCores Function

Copies all double cores of NFRecord to the specified array.

```
NResult N_API NFRecordGetDoubleCores(
    HNFRecord hRecord,
    NFDoubleCore * arDoubleCores
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>arDoubleCores</i>	[out] Pointer to array of NfDoubleCore that receives double cores.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>arDoubleCores</i> is NULL .

Remarks

Array *arDoubleCores* points to must be large enough to receive all NfRecord double cores. See [NfRecordGetDoubleCoreCount](#) function.

See Also

[NfRecord Module](#) | [HNfRecord](#) | [NfDoubleCore](#) | [NfRecordGetDoubleCoreCount](#)

6.5.2.39. NfRecordGetG Function

Retrieves the G of the NfRecord.

```
NResult N_API NfRecordGetG(
    HNfRecord hRecord,
    NByte * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>pValue</i>	[out] Pointer to NByte that receives G.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

G is a global fingerprint feature that reflects ridge density. It can have values from 0 to 255, so it occupies one byte. The bigger is value the bigger is ridge density.

See Also

[NfRecord Module](#) | [HNfRecord](#) | [NfRecordSetG](#)

6.5.2.40. NFRecordGetGMem Function

Retrieves the G of the packed NFRecord.

```
NResult N_API NFRecordGetGMem(
    const void * buffer,
    NSizeType bufferSize,
    NByte * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.
<i>pValue</i>	[out] Pointer to NByte that receives G.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

See Also

[NFRecord Module](#)

6.5.2.41. NFRecordGetHeight Function

Retrieves the height of the image the NFRecord is made from.

```
NResult N_API NFRecordGetHeight(
    HNFRecord hRecord,
    NUShort * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NUShort that receives height of fingerprint image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#)

6.5.2.42. NFRecordGetHeightMem Function

Retrieves the height of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetHeightMem(
    const void * buffer,
    NSizeType bufferSize,
    NUShort * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.
<i>pValue</i>	[out] Pointer to NUShort that receives height of fingerprint image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns 1 for version 1.0 format.

See Also

[NFRecord Module](#)

6.5.2.43. NFRecordGetHorzResolution Function

Retrieves the horizontal resolution in dpi of the image the NFRecord is made from.

```
NResult N_API NFRecordGetHorzResolution(
    HNFRecord hRecord,
    NUShort * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NUShort that receives horizontal resolution in pixels per inch of fingerprint image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#)

6.5.2.44. NFRecordGetHorzResolutionMem Function

Retrieves the horizontal resolution of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetHorzResolutionMem(
    const void * buffer,
    NSizeType bufferSize,
    NUShort * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.
<i>pValue</i>	[out] Pointer to NUShort that receives horizontal resolution in pixels per inch of fingerprint image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns 500 for version 1.0 format.

See Also

[NFRecord Module](#)

6.5.2.45. NFRecordGetImpressionType Function

Retrieves the impression type of the NFRecord.

```
NResult N_API NFRecordGetImpressionType(
    HNFRecord hRecord,
    NFImpressionType * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NFImpressionType that receives impression type.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFImpressionType](#) | [NFRecordSetImpressionType](#)

6.5.2.46. NFRecordGetImpressionTypeMem Function

Retrieves the impression type of the packed NFRecord.

```
NResult N_API NFRecordGetImpressionTypeMem(
    const void * buffer,
    NSizeType bufferSize,
    NFImpressionType * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NfRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NfRecord.
<i>pValue</i>	[out] Pointer to NFImpressionType that receives impression type.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NfRecord format.

Remarks

This function supports both NfRecord version 1.0 and 2.0 formats.

Always returns [nfitLiveScanPlain](#) for version 1.0 format.

See Also

[NfRecord Module](#) | [NFImpressionType](#)

6.5.2.47. NfRecordGetMaxSize Function

Retrieves the maximal size of a packed NfRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.

```
NResult N_API NfRecordGetMaxSize(
    NfMinutiaFormat minutiaFormat,
    NInt minutiaCount,
    NfRidgeCountsType ridgeCountsType,
    NInt coreCount,
    NInt deltaCount,
    NInt doubleCoreCount,
    NInt boWidth,
    NInt boHeight,
    NSizeType * pSize
);
```

Parameters

<i>minutiaFormat</i>	[in] The minutia format.
<i>minutiaCount</i>	[in] The number of minutiae.
<i>ridgeCountsType</i>	[in] The type of ridge counts.

<i>coreCount</i>	[in] The number of cores.
<i>deltaCount</i>	[in] The number of deltas.
<i>doubleCoreCount</i>	[in] The number of double cores.
<i>boWidth</i>	[in] The width of blocked orientations.
<i>boHeight</i>	[in] The height of blocked orientations.
<i>pSize</i>	[out] Pointer to NSizeType that receives maximal size of packed NRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>minutiaFormat</i> is invalid. - or - <i>ridgeCountsType</i> is invalid.
N_E_ARGUMENT_NULL	<i>pSize</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>minutiaCount</i> is less than zero or greater than or equal to NFR_MAX_MINUTIA_COUNT . - or - <i>coreCount</i> is less than zero or greater than or equal to NFR_MAX_CORE_COUNT . - or - <i>deltaCount</i> is less than zero or greater than or equal to NFR_MAX_DELTA_COUNT . - or - <i>doubleCoreCount</i> is less than zero or greater than or equal to NFR_MAX_DOUBLE_CORE_COUNT . - or - <i>boWidth</i> or <i>boHeight</i> is less than zero or greater than or equal to NFR_MAX_BLOCKED_ORIENTS_DIMENSION .

Remarks

This is a low-level function and can be changed in future version of the library.

The function calculates current (2.0) version packed size of NRecord.

boWidth and *boHeight* parameters are for compatibility only. If one of them or both is zero, blocked orientations are ignored.

See Also

[NFRecord Module](#) | [NFMinutiaFormat](#) | [NFMinutia](#) | [NFRidgeCountsType](#) | [NFCore](#) | [NFDelta](#) | [NFDoubleCore](#) | [NFRecordSaveToMemory](#)

6.5.2.48. NFRecordGetMaxSizeV1 Function

Retrieves the maximal size of a packed in version 1.0 format NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.

```
NResult N_API NFRecordGetMaxSizeV1(
    NFMinutiaFormat minutiaFormat,
    NInt minutiaCount,
    NInt coreCount,
    NInt deltaCount,
    NInt doubleCoreCount,
    NInt boWidth,
    NInt boHeight,
    NSizeType * pSize
);
```

Parameters

<i>minutiaFormat</i>	[in] The minutia format.
<i>minutiaCount</i>	[in] The number of minutiae.
<i>coreCount</i>	[in] The number of cores.
<i>deltaCount</i>	[in] The number of deltas.
<i>doubleCoreCount</i>	[in] The number of double cores.
<i>boWidth</i>	[in] The width of blocked orientations.
<i>boHeight</i>	[in] The height of blocked orientations.
<i>pSize</i>	[out] Pointer to NSizeType that receives maximal size of packed NFRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>minutiaFormat</i> is invalid.
N_E_ARGUMENT_NULL	<i>pSize</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>minutiaCount</i> is less than zero or greater than or equal to NFR_MAX_MINUTIA_COUNT . - or - <i>coreCount</i> is less than zero or greater than or equal to NFR_MAX_CORE_COUNT . - or - <i>deltaCount</i> is less than zero or greater than or equal to NFR_MAX_DELTA_COUNT . - or -

Error Code	Condition
	<p><i>doubleCoreCount</i> is less than zero or greater than or equal to NFR_MAX_DOUBLE_CORE_COUNT.</p> <p>- or -</p> <p><i>boWidth</i> or <i>boHeight</i> is less than zero or greater than or equal to NFR_MAX_BLOCKED_ORIENTS_DIMENSION.</p>

Remarks

This is a low-level function and can be changed in future version of the library.

boWidth and *boHeight* parameters are for compatibility only. If one of them or both is zero, blocked orientations are ignored.

See Also

[NFRecord Module](#) | [NFMinutiaFormat](#) | [NFMinutia](#) | [NFCore](#) | [NFDelta](#) | [NFDoubleCore](#) | [NFRecord-SaveToMemoryV1](#)

6.5.2.49. NFRecordGetMinutia Function

Retrieves the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordGetMinutia(
    HNFRecord hRecord,
    NInt index,
    NFMinutia * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of minutia to retrieve.
<i>pValue</i>	[out] Pointer to NFMinutia that receives minutia.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFMinutia](#) | [NFRecordGetMinutiaCount](#) | [NFRecordGetMinutiae](#)

6.5.2.50. NFRecordGetMinutiaCapacity Function

Retrieves the number of minutiae that the NFRecord can contain.

```
NResult N_API NFRecordGetMinutiaCapacity(
    HNFRecord hRecord,
    NInt * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NInt that receives number of minutiae NFRecord can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

Minutia capacity is the number of minutiae that the NFRecord can store. Minutia count (see [NFRecordGetMinutiaCount](#) function) is the number of minutiae that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordSetMinutiaCapacity](#) | [NFRecordGetMinutiaCount](#)

6.5.2.51. NFRecordGetMinutiaCount Function

Retrieves the number of minutiae in the NFRecord.

```
NResult N_API NFRecordGetMinutiaCount(
    HNFRecord hRecord,
    NInt * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NInt that receives number of minutiae.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

Minutia capacity (see [NFRecordGetMinutiaCapacity](#) and [NFRecordSetMinutiaCapacity](#) functions) is the number of minutiae that the NFRecord can store. Minutia count is the number of minutiae that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetMinutiaCapacity](#) | [NFRecordSetMinutiaCapacity](#)

6.5.2.52. NFRecordGetMinutiaFormat Function

Retrieves the format of the minutiae in NFRecord.

```
NResult N_API NFRecordGetMinutiaFormat(
    HNFRecord hRecord,
    NFMinutiaFormat * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NFMinutiaFormat that receives format of minutiae in the NFRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFMinutiaFormat](#) | [NFRecordSetMinutiaFormat](#)

6.5.2.53. NFRecordGetMinutiaNeighbor Function

Retrieves the minutia neighbor at the specified index of the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordGetMinutiaNeighbor(
    HNFRecord hRecord,
    NInt minutiaIndex,
    NInt index,
    NFMinutiaNeighbor * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
<i>minutiaIndex</i>	[in] The index of minutia.
<i>index</i>	[in] Index of minutia neighbor to retrieve.
<i>pValue</i>	[out] Pointer to NMinutiaNeighbor that receives minutia neighbor.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>minutiaIndex</i> is less than zero or greater than or equal to minutia count obtained using NRecordGetMinutiaCount function. - or - <i>index</i> is less than zero or greater than or equal to minutia neighbor count obtained using NRecordGetMinutiaNeighborCount function.

See Also

[NRecord Module](#) | [HNRecord](#) | [NMinutiaNeighbor](#) | [NRecordGetMinutiaCount](#) | [NRecordGetMinutiaNeighborCount](#) | [NRecordSetMinutiaNeighbor](#)

6.5.2.54. NRecordGetMinutiaNeighborCount Function

Retrieves the number of minutia neighbors in the minutia at the specified index of the NRecord.

```
NResult N_API NRecordGetMinutiaNeighborCount(
    HNRecord hRecord,
    NInt minutiaIndex,
    NInt * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
<i>minutiaIndex</i>	[in] The index of minutia.
<i>pValue</i>	[out] Pointer to NInt that receives number of minutia neighbors.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

[NFRecord Module](#) | [HNFRecord](#)

6.5.2.55. NFRecordGetMinutiaNeighbors Function

Copies all minutia neighbors of the minutia at the specified index of the NFRecord to the specified array.

```
NResult N_API NFRecordGetMinutiaNeighbors(
    HNFRecord hRecord,
    NInt minutiaIndex,
    NFMinutiaNeighbor * arMinutiaNeighbors
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>minutiaIndex</i>	[in] The index of minutia.
<i>arMinutiaNeighbors</i>	[out] Pointer to array of NFMinutiaNeighbor that receives minutia neighbors.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>arMinutiaNeighbors</i> is NULL .

Remarks

Array *arMinutiaNeighbors* points to must be large enough to receive all minutia neighbors. See [NFRecordGetMinutiaNeighborCount](#) function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFMinutiaNeighbor](#) | [NFRecordGetMinutiaCount](#) | [NFRecordGetMinutiaNeighborCount](#)

6.5.2.56. NFRecordGetMinutiae Function

Copies all minutiae of NFRecord to the specified array.

```
NResult N_API NFRecordGetMinutiae(
    HNFRecord hRecord,
    NFMinutia * arMinutiae
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>arMinutiae</i>	[out] Pointer to array of NFMinutia that receives minutiae.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>arMinutiae</i> is NULL .

Remarks

Array *arMinutiae* points to must be large enough to receive all NfRecord minutiae. See [NFRecordGetMinutiaCount](#) function.

See Also

[NFRecord Module](#) | [HNfRecord](#) | [NFMinutia](#) | [NFRecordGetMinutiaCount](#)

6.5.2.57. NFRecordGetPatternClass Function

Retrieves the pattern class of the NfRecord.

```
NResult N_API NFRecordGetPatternClass(
    HNfRecord hRecord,
    NFPatternClass * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>pValue</i>	[out] Pointer to NFPatternClass that receives fingerprint pattern class.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

Remarks

This function supports both NfRecord version 1.0 and 2.0 formats.

Always returns [nfpcUnknown](#) for version 1.0 format.

See Also

[NFRecord Module](#) | [HNFRRecord](#) | [NFPatternClass](#) | [NFRecordSetPatternClass](#)

6.5.2.58. NFRecordGetPatternClassMem Function

Retrieves the pattern class of the packed NFRecord.

```
NResult N_API NFRecordGetPatternClassMem(
    const void * buffer,
    NSizeType bufferSize,
    NFPatternClass * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.
<i>pValue</i>	[out] Pointer to NFPatternClass that receives fingerprint pattern class.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns [nfpcUnknown](#) for version 1.0 format.

See Also

[NFRecord Module](#) | [NFPatternClass](#)

6.5.2.59. NFRecordGetPosition Function

Retrieves the finger position of the NFRecord.

```
NResult N_API NFRecordGetPosition(
    HNFRRecord hRecord,
    NFPosition * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>pValue</i>	[out] Pointer to NFPosition that receives finger position.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NfRecord Module](#) | [HNfRecord](#) | [NFPosition](#) | [NfRecordSetPosition](#)

6.5.2.60. NfRecordGetPositionMem Function

Retrieves the finger position of the packed NfRecord.

```
NResult N_API NfRecordGetPositionMem(
    const void * buffer,
    NSizeType bufferSize,
    NFPosition * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NfRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NfRecord.
<i>pValue</i>	[out] Pointer to NFPosition that receives finger position.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NfRecord format.

Remarks

This function supports both NfRecord version 1.0 and 2.0 formats.

Always returns [nfpUnknown](#) for version 1.0 format.

See Also

[NFRecord Module](#) | [NFPosition](#)

6.5.2.61. NFRecordGetQuality Function

Retrieves the quality of the NFRecord.

```
NResult N_API NFRecordGetQuality(
    HNFRecord hRecord,
    NByte * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NByte that receives fingerprint quality.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordSetQuality](#)

6.5.2.62. NFRecordGetQualityMem Function

Retrieves the quality of the packed NFRecord.

```
NResult N_API NFRecordGetQualityMem(
    const void * buffer,
    NSizeType bufferSize,
    NByte * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.
<i>pValue</i>	[out] Pointer to NByte that receives fingerprint quality.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NfRecord format.

Remarks

This function supports both NfRecord version 1.0 and 2.0 formats.

Always returns 0 for version 1.0 format.

See Also

[NfRecord Module](#)

6.5.2.63. NfRecordGetRidgeCountsType Function

Retrieves the ridge counts type the NfRecord contains.

```
NResult N_API NfRecordGetRidgeCountsType(
    HNFRecord hRecord,
    NFRidgeCountsType * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>pValue</i>	[out] Pointer to NFRidgeCountsType that receives ridge counts type stored in NfRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NfRecord Module](#) | [HNFRecord](#) | [NFRidgeCountsType](#) | [NfRecordSetRidgeCountsType](#)

6.5.2.64. NfRecordGetSize Function

Calculates packed size of the NfRecord.

```
NResult N_API NfRecordGetSize(
    HNFRecord hRecord,
    NUInt flags,
    NSizeType * pSize
);
```


Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NfRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pSize</i> is NULL .

Remarks

The function calculates current (2.0) version packed size of NfRecord.

For the list of flags that are supported see [NfRecordSaveToMemory](#) function.

See Also

[NfRecord Module](#) | [HNfRecord](#) | [NfRecordSaveToMemory](#)

6.5.2.65. NfRecordGetSizeV1 Function

Calculates packed in version 1.0 format size of the NfRecord.

```
NResult N_API NfRecordGetSizeV1(
    HNfRecord hRecord,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NfRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pSize</i> is NULL .

Remarks

For the list of flags that are supported see [NFRecordSaveToMemoryV1](#) function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordSaveToMemoryV1](#)

6.5.2.66. NFRecordGetVertResolution Function

Retrieves the vertical resolution of the image the NFRecord is made from.

```
NResult N_API NFRecordGetVertResolution(
    HNFRecord hRecord,
    NUShort * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>pValue</i>	[out] Pointer to NUShort that receives vertical resolution in pixels per inch of fingerprint image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#)

6.5.2.67. NFRecordGetVertResolutionMem Function

Retrieves the vertical resolution of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetVertResolutionMem(
    const void * buffer,
    NSizeType bufferSize,
    NUShort * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.
<i>pValue</i>	[out] Pointer to NUShort that receives vertical resolution in pixels per inch of fingerprint image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NRecord format.

Remarks

This function supports both NRecord version 1.0 and 2.0 formats.

Always returns 500 for version 1.0 format.

See Also

[NRecord Module](#)

6.5.2.68. NRecordGetWidth Function

Retrieves the width of the image the NRecord is made from.

```
NResult N_API NRecordGetWidth(
    HNRecord hRecord,
    NUShort * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
<i>pValue</i>	[out] Pointer to NUShort that receives width of fingerprint image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NRecord Module](#) | [HNRecord](#)

6.5.2.69. NRecordGetWidthMem Function

Retrieves the width of the image the packed NRecord is made from.

```
NResult N_API NRecordGetWidthMem(
```

```

    const void * buffer,
    NSizeType bufferSize,
    NUShort * pValue
);

```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.
<i>pValue</i>	[out] Pointer to NUShort that receives width of fingerprint image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns 1 for version 1.0 format.

See Also

[NFRecord Module](#)

6.5.2.70. NFRecordInsertCore Function

Inserts a [NFCore](#) into the NFRecord at the specified index.

```

NResult N\_API NFRecordInsertCore(
    HNFRecord hRecord,
    NInt index,
    const NFCore * pValue
);

```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index at which core is inserted.
<i>pValue</i>	[in] Pointer to the NFCore to insert.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than core count obtained using NFRecordGetCoreCount function.
N_E_INVALID_OPERATION	Number of cores in NFRecord (see NFRecordGetCoreCount) is equal to NFR_MAX_CORE_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFCore](#) | [NFRecordGetCoreCount](#)

6.5.2.71. NFRecordInsertDelta Function

Inserts a [NFDelta](#) into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertDelta(
    HNFRecord hRecord,
    NInt index,
    const NFDelta * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index at which delta is inserted.
<i>pValue</i>	[in] Pointer to the NFDelta to insert.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than delta count obtained using NFRecordGetDeltaCount function.
N_E_INVALID_OPERATION	Number of deltas in NFRecord (see NFRecordGetDeltaCount) is equal to NFR_MAX_DELTA_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFDelta](#) | [NFRecordGetDeltaCount](#)

6.5.2.72. NFRecordInsertDoubleCore Function

Inserts a [NFDoubleCore](#) into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertDoubleCore(
    HNFRecord hRecord,
    NInt index,
    const NFDoubleCore * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index at which double core is inserted.
<i>pValue</i>	[in] Pointer to the NFDoubleCore to insert.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than double core count obtained using NFRecordGetDoubleCoreCount function.
N_E_INVALID_OPERATION	Number of double core in NFRecord (see NFRecordGetDoubleCoreCount) is equal to NFR_MAX_DOUBLE_CORE_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFDoubleCore](#) | [NFRecordGetDoubleCoreCount](#)

6.5.2.73. NFRecordInsertMinutia Function

Inserts a [NFMinutia](#) into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertMinutia(
    HNFRecord hRecord,
    NInt index,
    const NFMinutia * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index at which minutia is inserted.

<i>pValue</i>	[in] Pointer to the NFMinutia to insert.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than minutia count obtained using NFRecordGetMinutiaCount function.
N_E_INVALID_OPERATION	Number of minutia in NFRecord (see NFRecordGetMinutiaCount) is equal to NFR_MAX_MINUTIA_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFMinutia](#) | [NFRecordGetMinutiaCount](#)

6.5.2.74. NFRecordRemoveCore Function

Removes the core at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveCore(
    HNFRecord hRecord,
    NInt index
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of core to remove.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetCoreCount](#)

6.5.2.75. NFRecordRemoveDelta Function

Removes the delta at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveDelta(
    HNFRecord hRecord,
    NInt index
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of delta to remove.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetDeltaCount](#)

6.5.2.76. NFRecordRemoveDoubleCore Function

Removes the double core at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveDoubleCore(
    HNFRecord hRecord,
    NInt index
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of double core to remove.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to

Error Code	Condition
	double core count obtained using NFRecordGetDoubleCoreCount function.

See Also

[NFRecord Module](#) | [HNRecord](#) | [NFRecordGetDoubleCoreCount](#)

6.5.2.77. NFRecordRemoveMinutia Function

Removes the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveMinutia(
    HNRecord hRecord,
    NInt index
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of minutia to remove.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

[NFRecord Module](#) | [HNRecord](#) | [NFRecordGetMinutiaCount](#)

6.5.2.78. NFRecordSaveToMemory Function

Packs the NFRecord into the specified memory buffer.

```
NResult N_API NFRecordSaveToMemory(
    HNRecord hRecord,
    void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
----------------	-------------------------------------

<i>buffer</i>	[out] Pointer to memory buffer to store packed NRecord. Can be NULL .
<i>bufferSize</i>	[in] Size of memory buffer to store packed NRecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>buffer</i> is not NULL and <i>bufferSize</i> is less than size required to store packed NRecord.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pSize</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferSize</i> is not zero.

Remarks

The function packs NRecord in current (2.0) version format.

If *buffer* is [NULL](#) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as [NRecordGetSize](#) function.

If *buffer* is not [NULL](#), *bufferSize* must not be less than value calculated with [NRecordGetSize](#) function.

Note that blocked orientations are not packed by default.

The following flags are supported:

- [NFR_SAVE_BLOCKED_ORIENTS](#)
- [NFR_SKIP_CURVATURES](#)
- [NFR_SKIP_GS](#)
- [NFR_SKIP_QUALITIES](#)
- [NFR_SKIP RIDGE_COUNTS](#)
- [NFR_SKIP_SINGULAR_POINTS](#)

See Also

[NRecord Module](#) | [HNRecord](#) | [NRecordGetSize](#) | [NRecordCreateFromMemory](#)

6.5.2.79. NRecordSaveToMemoryV1 Function

Packs the NRecord into the specified memory buffer in version 1.0 format.

```
NResult N_API NRecordSaveToMemoryV1(
    HNRecord hRecord,
    void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NSizeType * pSize
```

```
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NRecord. Can be NULL .
<i>bufferSize</i>	[in] Size of memory buffer to store packed NRecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>buffer</i> is not NULL and <i>bufferSize</i> is less than size required to store packed NRecord.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pSize</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is [NULL](#) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as [NRecordGetSizeV1](#) function.

If *buffer* is not [NULL](#), *bufferSize* must not be less than value calculated with [NRecordGetSizeV1](#) function.

Note that blocked orientations are not packed by default.

The following flags are supported:

- [NFR_SAVE_BLOCKED_ORIENTS](#)
- [NFR_SKIP_CURVATURES](#)
- [NFR_SKIP_GS](#)
- [NFR_SKIP_SINGULAR_POINTS](#)

See Also

[NRecord Module](#) | [HNRecord](#) | [NRecordCreateFromMemory](#) | [NRecordGetSizeV1](#)

6.5.2.80. NRecordSetCbeffProductType Function

Sets the Cbeff product type.

```
NResult N_API NRecordSetCbeffProductType(
    HNRecord hRecord,
    NUShort value
```

```
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>value</i>	[in] Cbeff product type.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NfRecord Module](#) | [HNfRecord](#) | [NfRecordGetCbeffProductType](#) [NfRecordGetCbeffProductTypeMem](#)

6.5.2.81. NfRecordSetCore Function

Sets a [NfCore](#) at the specified index of the NfRecord.

```
NResult N_API NfRecordSetCore(
    HNfRecord hRecord,
    NInt index,
    const NfCore * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NfRecord object.
<i>index</i>	[in] Index of core to set.
<i>pValue</i>	[in] Pointer to the NfCore to set.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to core count obtained using NfRecordGetCoreCount function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFCore](#) | [NFRecordGetCoreCount](#) | [NFRecordGetCore](#)

6.5.2.82. NFRecordSetCoreCapacity Function

Sets the number of cores that the NFRecord can contain.

```
NResult N_API NFRecordSetCoreCapacity(
    HNFRecord hRecord,
    NInt value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New number of cores NFRecord can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>value</i> is less than core count obtained using NFRecordGetCoreCount function.

Remarks

Core capacity is the number of cores that the NFRecord can store. Core count (see [NFRecordGetCoreCount](#) function) is the number of cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetCoreCapacity](#) | [NFRecordGetCoreCount](#)

6.5.2.83. NFRecordSetDelta Function

Sets a [NFDelta](#) at the specified index of the NFRecord.

```
NResult N_API NFRecordSetDelta(
    HNFRecord hRecord,
    NInt index,
    const NFDelta * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of delta to set.
<i>pValue</i>	[in] Pointer to the NFDelta to set.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount function.

See Also

[NFRecord Module](#) | [HNRecord](#) | [NFDelta](#) | [NFRecordGetDeltaCount](#) | [NFRecordGetDelta](#)

6.5.2.84. NFRecordSetDeltaCapacity Function

Sets the number of deltas that the NFRecord can contain.

```
NResult N_API NFRecordSetDeltaCapacity(
    HNRecord hRecord,
    NInt value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New number of deltas NFRecord can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>value</i> is less than delta count obtained using NFRecordGetDeltaCount function.

Remarks

Delta capacity is the number of deltas that the NFRecord can store. Delta count (see [NFRecordGetDeltaCount](#) function) is the number of deltas that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNRecord](#) | [NFRecordGetDeltaCapacity](#) | [NFRecordGetDeltaCount](#)

6.5.2.85. NFRecordSetDoubleCore Function

Sets a [NFDoubleCore](#) at the specified index of the NFRecord.

```
NResult N_API NFRecordSetDoubleCore(
    HNFRecord hRecord,
    NInt index,
    const NFDoubleCore * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of double core to set.
<i>pValue</i>	[in] Pointer to the NFDoubleCore to set.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to double core count obtained using NFRecordGetDoubleCoreCount function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFDoubleCore](#) | [NFRecordGetDoubleCoreCount](#) | [NFRecordGetDoubleCore](#)

6.5.2.86. NFRecordSetDoubleCoreCapacity Function

Sets the number of double cores that the NFRecord can contain.

```
NResult N_API NFRecordSetDoubleCoreCapacity(
    HNFRecord hRecord,
    NInt value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New number of double cores NFRecord can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>value</i> is less than double core count obtained using NFRecordGetDoubleCoreCount function.

Remarks

Double core capacity is the number of double cores that the NFRecord can store. Double core count (see [NFRecordGetDoubleCoreCount](#) function) is the number of double cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding double cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetDoubleCoreCapacity](#) | [NFRecordGetDoubleCoreCount](#)

6.5.2.87. NFRecordSetG Function

Sets the G of the NFRecord.

```
NResult N_API NFRecordSetG(
    HNFRecord hRecord,
    NByte value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New G value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

Remarks

G is a global fingerprint feature that reflects ridge density. It can have values from 0 to 255, so it occupies one byte. The bigger is value the bigger is ridge density.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetG](#)

6.5.2.88. NFRecordSetImpressionType Function

Sets the impression type of the NFRecord.

```
NResult N_API NFRecordSetImpressionType(
```



```

    HNFRecord hRecord,
    NFImpressionType value
);

```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New impression type value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>value</i> is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFImpressionType](#) | [NFRecordGetImpressionType](#)

6.5.2.89. NFRecordSetMinutia Function

Sets a [NFMinutia](#) at the specified index of the NFRecord.

```

NResult N_API NFRecordSetMinutia(
    HNFRecord hRecord,
    NInt index,
    const NFMinutia * pValue
);

```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>index</i>	[in] Index of minutia to set.
<i>pValue</i>	[in] Pointer to the NFMinutia to set.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFMinutia](#) | [NFRecordGetMinutiaCount](#) | [NFRecordGetMinutia](#)

6.5.2.90. NFRecordSetMinutiaCapacity Function

Sets the number of minutiae that the NFRecord can contain.

```
NResult N_API NFRecordSetMinutiaCapacity(
    HNFRecord hRecord,
    NInt value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New number of minutiae NFRecord can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>value</i> is less than minutia count obtained using NFRecordGetMinutiaCount function.

Remarks

Minutia capacity is the number of minutiae that the NFRecord can store. Minutia count (see [NFRecordGetMinutiaCount](#) function) is the number of minutiae that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetMinutiaCapacity](#) | [NFRecordGetMinutiaCount](#)

6.5.2.91. NFRecordSetMinutiaFormat Function

Sets the format of the minutiae in NFRecord.

```
NResult N_API NFRecordSetMinutiaFormat(
    HNFRecord hRecord,
    NFMinutiaFormat value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New minutia format value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>value</i> is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFMinutiaFormat](#) | [NFRecordGetMinutiaFormat](#)

6.5.2.92. NFRecordSetMinutiaNeighbor Function

Sets a [NFMinutiaNeighbor](#) at the specified index of the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordSetMinutiaNeighbor(
    HNFRecord hRecord,
    NInt minutiaIndex,
    NInt index,
    const NFMinutiaNeighbor * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>minutiaIndex</i>	[in] The index of minutia.
<i>index</i>	[in] Index of minutia neighbor to set.
<i>pValue</i>	[in] Pointer to the NFMinutiaNeighbor to set.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>minutiaIndex</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function. - or - <i>index</i> is less than zero or greater than or equal to minutia neighbor count obtained using NFRecordGetMinutiaNeighborCount function.

See Also

[NFRecord Module](#) | [HNFRRecord](#) | [NFMinutiaNeighbor](#) | [NFRecordGetMinutiaCount](#) | [NFRecordGetMinutiaNeighborCount](#) | [NFRecordGetMinutiaNeighbor](#)

6.5.2.93. NFRecordSetPatternClass Function

Sets the pattern class of the NFRecord.

```
NResult N_API NFRecordSetPatternClass(
    HNFRRecord hRecord,
    NFPatternClass value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New fingerprint pattern class value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>value</i> is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRRecord](#) | [NFPatternClass](#) | [NFRecordGetPatternClass](#)

6.5.2.94. NFRecordSetPosition Function

Sets the finger position of the NFRecord.

```
NResult N_API NFRecordSetPosition(
    HNFRRecord hRecord,
    NFPosition value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New finger position value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>value</i> is invalid.

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFPosition](#) | [NFRecordGetPosition](#)

6.5.2.95. NFRecordSetQuality Function

Sets the quality of the NFRecord.

```
NResult N_API NFRecordSetQuality(
    HNFRecord hRecord,
    NByte value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New fingerprint quality value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordGetQuality](#)

6.5.2.96. NFRecordSetRidgeCountsType Function

Sets the ridge counts type the NFRecord contains.

```
NResult N_API NFRecordSetRidgeCountsType(
    HNFRecord hRecord,
    NFRidgeCountsType value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>value</i>	[in] New ridge counts type value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>value</i> is invalid.
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRidgeCountsType](#) | [NFRecordGetRidgeCountsType](#)

6.5.2.97. NFRecordSortMinutiae Function

Sorts minutiae in NFRecord by the specified order.

```
NResult N_API NFRecordSortMinutiae(
    HNFRecord hRecord,
    NFMinutiaOrder order
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>order</i>	[in] Specifies minutia order.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value <i>order</i> is incorrect.
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFMinutiaOrder](#)

6.5.2.98. NFRecordTruncateMinutiae Function

Truncates minutiae in NFRecord by peeling off the minutiae convex hull while minutia count is greater than the specified maximal count.

```
NResult N_API NFRecordTruncateMinutiae(
    HNFRecord hRecord,
    NInt maxCount
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>maxCount</i>	Maximal minutia count to be present in the NFRecord after truncation.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>maxCount</i> is less than zero or greater than NFR_MAX_MINUTIA_COUNT .

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordAddMinutia](#)

6.5.2.99. NFRecordTruncateMinutiaeByQuality Function

Truncates minutiae in NFRecord by removing minutiae which NFMinutia.Quality field value is less than the specified threshold while minutia count is greater than the specified maximal count.

```
NResult N_API NFRecordTruncateMinutiaeByQuality(
    HNFRecord hRecord,
    NByte threshold,
    NInt maxCount
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NFRecord object.
<i>threshold</i>	Specifies minimal NFMinutia.Quality field value of minutiae not to be removed.
<i>maxCount</i>	Maximal minutia count to be present in the NFRecord after truncation.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	<i>maxCount</i> is less than zero or greater than NFR_MAX_MINUTIA_COUNT .
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>threshold</i> is incorrect.
N_E_INVALID_OPERATION	Minutia format is not NFMinutiaFormat.nfmfHasQuality .

Remarks

Minutia count after truncation may be greater than *maxCount* if there is no enough minutiae with quality less than *threshold*.

See Also

[NFRecord Module](#) | [HNFRecord](#) | [NFRecordAddMinutia](#)

6.5.2.100. NFRidgeCountsType Enumeration

Specifies the type of ridge counts contained in a NFRecord.

```
typedef enum NFRidgeCountsType_ { } NFRidgeCountsType;
```

Members

<code>nfrctEightNeighbors</code>	The NFRecord contains ridge counts to closest minutia in each of the eight sectors of each minutia. First sector starts at minutia angle.
<code>nfrctEightNeighborsWithIndexes</code>	The NFRecord contains ridge counts to eight neighbors of each minutia.
<code>nfrctFourNeighbors</code>	The NFRecord contains ridge counts to closest minutia in each of the four sectors of each minutia. First sector starts at minutia angle.
<code>nfrctFourNeighborsWithIndexes</code>	The NFRecord contains ridge counts to four neighbors of each minutia.
<code>nfrctNone</code>	The NFRecord does not contain ridge counts.
<code>nfrctUnspecified</code>	For internal use.

Remarks

Extracted template with `nfrctEightNeighborsWithIndexes` parameter is bigger than the template extracted with `nfrctEightNeighbors` parameter. Templates extracted with `nfrctEightNeighborsWithIndexes` parameter is faster than the templates extracted with `nfrctEightNeighbors` parameter.

Extracted template with `nfrctFourNeighborsWithIndexes` parameter is bigger than the template extracted with `nfrctFourNeighbors` parameter. Templates extracted with `nfrctFourNeighborsWithIndexes` parameter is faster than the templates extracted with `nfrctFourNeighbors` parameter.

See Also

[NFRecord Module](#)

6.5.3. NFTemplate Module

Provides functionality for packing, unpacking and editing Neurotechnology Fingers Templates (NFTemplates).

Header file: `NFTemplate.h`

Functions

NFTemplateAddRecord	Adds an empty finger record to the NFTemplate.
NFTemplateAddRecordCopy	Adds a copy of the finger record to the NFTemplate.
NFTemplateAddRecordFromMemory	Unpacks a finger record from the specified memory buffer and adds it to the NFTemplate.
NFTemplateCalculateSize	Calculates the size of a packed NFTemplate containing the specified number of finger records of specified size.
NFTemplateCheck	Checks if format of the packed NFTemplate is correct.

NFTemplateClearRecords	Removes all finger records from the NFTemplate.
NFTemplateClone	Creates a copy of the NFTemplate.
NFTemplateCreate	Creates an empty NFTemplate.
NFTemplateCreateFromMemory	Unpacks a NFTemplate from the specified memory buffer.
NFTemplateFree	Deletes the NFTemplate. After the object is deleted the specified handle is no longer valid.
NFTemplateGetRecord	Retrieves the finger record at the specified index of the NFTemplate.
NFTemplateGetRecordCapacity	Retrieves the number of finger records that the NFTemplate can contain..
NFTemplateGetRecordCount	Retrieves the number of finger records in the NFTemplate.
NFTemplateGetRecordCountMem	Retrieves the number of finger records in the packed NFTemplate.
NFTemplateGetSize	Calculates packed size of the NFTemplate.
NFTemplateInfoDispose	For internal use.
NFTemplatePack	Packs packed finger records as NFTemplate into the specified memory buffer.
NFTemplateRemoveRecord	Removes the finger record at the specified index of the NFTemplate.
NFTemplateSaveToMemory	Packs the NFTemplate into the specified memory buffer.
NFTemplateSetRecordCapacity	Sets the number of finger records that the NFTemplate can contain.
NFTemplateUnpack	Unpacks packed finger records from the packed NFTemplate.

Structures

NFTemplateInfo	For internal use.
--------------------------------	-------------------

Types

HNFTemplate	Handle to NFTemplate object.
-----------------------------	------------------------------

Macros

NFT_MAX_RECORD_COUNT	The maximum number of finger records NFTemplate can contain.
NFT_PROCESS_FIRST_RECORD_ONLY	The flag indicating whether only the first finger record should be unpacked or packed while unpacking or packing NFTemplate.

See Also

[NTemplate Library](#)

6.5.3.1. NTemplateAddRecord Function

Adds an empty finger record to the NTemplate.

```
NResult N_API NTemplateAddRecord(
    HNFTemplate hTemplate,
    NUShort width,
    NUShort height,
    NUShort horzResolution,
    NUShort vertResolution,
    NUInt flags,
    HNFRecord * pHRecord
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NTemplate object.
<i>width</i>	[in] Specifies width of fingerprint image.
<i>height</i>	[in] Specifies height of fingerprint image.
<i>horzResolution</i>	[in] Specifies horizontal resolution in pixels per inch of fingerprint image.
<i>vertResolution</i>	[in] Specifies vertical resolution in pixels per inch of fingerprint image.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pHRecord</i>	[out] Pointer to HNFRecord that receives handle to created NRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is zero.
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHRecord</i> is NULL .
N_E_INVALID_OPERATION	Number of finger records in NTemplate (see NTemplateGetRecordCount) is equal to NFT_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see [NFRecordCreate](#) function.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [HNFRecord](#) | [NFTemplateGetRecordCount](#) | [NFRecordCreate](#)

6.5.3.2. NFTemplateAddRecordCopy Function

Adds a copy of the finger record to the NFTemplate.

```
NResult N_API NFTemplateAddRecordCopy(
    HNFTemplate hTemplate,
    HNFRecord hSrcRecord,
    HNFRecord * pHRecord
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
<i>hSrcRecord</i>	[in] Handle to the NFRecord object.
<i>pHRecord</i>	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>hSrcRecord</i> or <i>pHRecord</i> is NULL .
N_E_INVALID_OPERATION	Number of finger records in NFTemplate (see NFTemplateGetRecordCount) is equal to NFT_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [HNFRecord](#) | [NFTemplateGetRecordCount](#)

6.5.3.3. NFTemplateAddRecordFromMemory Function

Unpacks a finger record from the specified memory buffer and adds it to the NFTemplate.

```
NResult N_API NFTemplateAddRecordFromMemory(
    HNFTemplate hTemplate,
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    HNFRecord * pHRecord
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFRecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pHRecord</i>	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHRecord</i> , or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Number of finger records in NFTemplate (see NFTemplateGetRecordCount) is equal to NFT_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see [NFRecordCreateFromMemory](#) function.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [HNFRecord](#) | [NFTemplateGetRecordCount](#) | [NFRecordCreateFromMemory](#)

6.5.3.4. NFTemplateCalculateSize Function

Calculates the size of a packed NFTemplate containing the specified number of finger records of specified size.

```
NResult N_API NFTemplateCalculateSize(
    NInt recordCount,
    NSizeType * arRecordSizes,
    NSizeType * pSize
);
```

Parameters

<i>recordCount</i>	[in] The number of finger records.
<i>arRecordSizes</i>	[in] Pointer to array of NSizeType that contains packed sizes of each finger record.
<i>pSize</i>	[out] Pointer to NSizeType that receives calculated size

	of packed NFTemplate.
--	-----------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Any entry of the array <i>arRecordSizes</i> points to is less than minimal finger record size.
N_E_ARGUMENT_NULL	<i>arRecordSizes</i> or <i>pSize</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>recordCount</i> is less than zero or greater than NFT_MAX_RECORD_COUNT .

Remarks

This is a low-level function and can be changed in future version of the library.

Use [NFRecordGetMaxSize](#) function to calculate packed size of an individual finger record.

See Also

[NFTemplate Module](#) | [NFRecordGetMaxSize](#) | [NFTemplateSaveToMemory](#)

6.5.3.5. NFTemplateCheck Function

Checks if format of the packed NFTemplate is correct.

```
NResult N_API NFTemplateCheck(
    const void * buffer,
    NSizeType bufferSize
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFTemplate format.

See Also

[NFTemplate Module](#)

6.5.3.6. NFTemplateClearRecords Function

Removes all finger records from the NFTemplate.

```
NResult N_API NFTemplateClearRecords(
    HNFTemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
------------------	---------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

See Also

[NFTemplate Module](#) | [HNFTemplate](#)

6.5.3.7. NFTemplateClone Function

Creates a copy of the NFTemplate.

```
NResult N_API NFTemplateClone(
    HNFTemplate hTemplate,
    HNFTemplate * pHClonedTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
<i>pHClonedTemplate</i>	[out] Pointer to HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHClonedTemplate</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NFTemplateFree](#) function.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateFree](#)

6.5.3.8. NFTemplateCreate Function

Creates an empty NFTemplate.

```
NResult N_API NFTemplateCreate(
    HNFTemplate * pHTemplate
);
```

Parameters

<i>pHTemplate</i>	[out] Pointer to HNFTemplate that receives handle to created NFTemplate object.
-------------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHTemplate</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NFTemplateFree](#) function.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateFree](#)

6.5.3.9. NFTemplateCreateFromMemory Function

Unpacks a NFTemplate from the specified memory buffer.

```
NResult N_API NFTemplateCreateFromMemory(
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NFTemplateInfo * pInfo,
    HNFTemplate * pHTemplate
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFTemplate.

<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pInfo</i>	[out] For internal use. Must be NULL .
<i>pHTemplate</i>	[out] Pointer to HNFTemplate that receives handle to newly created NFTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHTemplate</i> or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFTemplate format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- [NFT_PROCESS_FIRST_RECORD_ONLY](#)
- Flags supported by [NFRecordCreateFromMemory](#) function are applied to each finger record contained in the NFTemplate.

Created object must be deleted using [NFTemplateFree](#) function.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateInfo](#) | [NFTemplateFree](#) | [NFRecordCreateFromMemory](#) | [NFTemplateSaveToMemory](#)

6.5.3.10. NFTemplateFree Function

Deletes the NFTemplate. After the object is deleted the specified handle is no longer valid.

```
void N\_API NFTemplateFree(
    HNFTemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
------------------	---------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

Remarks

If *hTemplate* is [NULL](#), does nothing.

See Also

[NFTemplate Module](#) | [HNFTemplate](#)

6.5.3.11. NFTemplateGetRecord Function

Retrieves the finger record at the specified index of the NFTemplate.

```
NResult N_API NFTemplateGetRecord(
    HNFTemplate hTemplate,
    NInt index,
    HNFRecord * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
<i>index</i>	[in] Index of finger record to retrieve.
<i>pValue</i>	[out] Pointer to HNFRecord that receives handle to NFRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to finger record count obtained using NFTemplateGetRecordCount function.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateGetRecordCount](#)

6.5.3.12. NFTemplateGetRecordCapacity Function

Retrieves the number of finger records that the NFTemplate can contain..

```
NResult N_API NFTemplateGetRecordCapacity(
    HNFTemplate hTemplate,
    NInt * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
<i>pValue</i>	[out] Pointer to NInt that receives number of finger records NFTemplate can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .

Remarks

Finger record capacity is the number of finger records that the NFTemplate can store. Finger record count (see [NFTemplateGetRecordCount](#) function) is the number of finger records that are actually in the NFTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateSetRecordCapacity](#) | [NFTemplateGetRecordCount](#)

6.5.3.13. NFTemplateGetRecordCount Function

Retrieves the number of finger records in the NFTemplate.

```
NResult N_API NFTemplateGetRecordCount(
    HNFTemplate hTemplate,
    NInt * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to NFTemplate object.
<i>pValue</i>	[out] Pointer to NInt that receives number of finger records.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .

Remarks

Finger record capacity (see [NFTemplateGetRecordCapacity](#) and [NFTemplateSetRecordCapacity](#) func-

tions) is the number of finger records that the NFTemplate can store. Finger record count is the number of finger records that are actually in the NFTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateGetRecordCapacity](#) | [NFTemplateSetRecordCapacity](#)

6.5.3.14. NFTemplateGetRecordCountMem Function

Retrieves the number of finger records in the packed NFTemplate.

```
NResult N_API NFTemplateGetRecordCountMem(
    const void * buffer,
    NSizeType bufferSize,
    NInt * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFTemplate.
<i>pValue</i>	[out] Pointer to NInt that receives number of finger records.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFTemplate format.

See Also

[NFTemplate Module](#)

6.5.3.15. NFTemplateGetSize Function

Calculates packed size of the NFTemplate.

```
NResult N_API NFTemplateGetSize(
    HNFTemplate hTemplate,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hTemplate</i>	[in] Handle to NFTemplate object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives calculated size of packed NFTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pSize</i> is NULL .

Remarks

For the list of flags that are supported see [NFTemplateSaveToMemory](#) function.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateSaveToMemory](#)

6.5.3.16. NFTemplatePack Function

Packs packed finger records as NFTemplate into the specified memory buffer.

```
NResult N_API NFTemplatePack(
    NInt recordCount,
    const void * * arRecords,
    NSizeType * arRecordSizes,
    void * buffer,
    NSizeType bufferSize,
    NSizeType * pSize
);
```

Parameters

<i>recordCount</i>	[in] The number of finger records.
<i>arRecords</i>	[in] Pointer to array of void * that contains packed finger records.
<i>arRecordSizes</i>	[in] Pointer to array of NSizeType that contains packed sizes of each finger record.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NFTemplate.
<i>bufferSize</i>	[in] Size of memory buffer to store packed NFTemplate.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NFTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<p><i>bufferSize</i> is less than size required to store packed NFTemplate.</p> <p>- or -</p> <p>Any entry of the array <i>arRecordSizes</i> points to is less than minimal finger record size.</p> <p>- or -</p> <p>Any entry of the array <i>arRecords</i> points to is NULL.</p> <p>- or -</p> <p>Any entry of the array <i>arRecordSizes</i> points to is not equal to size stored in memory buffer according entry of the array <i>arRecords</i> points to, points to.</p>
N_E_ARGUMENT_NULL	<i>arRecords</i> or <i>buffer</i> or <i>pSize</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>recordCount</i> is less than zero or greater than NFT_MAX_RECORD_COUNT .
N_E_FORMAT	Data in memory buffer any entry of the array <i>arRecords</i> points to is inconsistent with NFRecord format.

Remarks

This is a low-level function and can be changed in future version of the library.

bufferSize must not be less than value calculated with [NFTemplateCalculateSize](#) function with the same parameter values.

See Also

[NFTemplate Module](#) | [NFTemplateCalculateSize](#) | [NFTemplateUnpack](#)

6.5.3.17. NFTemplateRemoveRecord Function

Removes the finger record at the specified index of the NFTemplate.

```
NResult N_API NFTemplateRemoveRecord(
    HNFTemplate hTemplate,
    NInt index
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
<i>index</i>	[in] Index of finger record to remove.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater or equal than records count.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateGetRecordCount](#)

6.5.3.18. NFTemplateSaveToMemory Function

Packs the NFTemplate into the specified memory buffer.

```
NResult N_API NFTemplateSaveToMemoryEx(
    HNFTemplate hTemplate,
    void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hTemplate</i>	[in] Handle to NFTemplate object.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NFTemplate. Can be NULL .
<i>bufferSize</i>	[in] Size of memory buffer to store packed NFTemplate.
<i>flags</i>	[in] Bitwise combination of zero or more flags that control behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NFTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>buffer</i> is not NULL and <i>bufferSize</i> is less than size required to store packed NFTemplate.
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pSize</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is [NULL](#) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the

same effect as [NFTemplateGetSize](#) function.

If *buffer* is not [NULL](#), *bufferSize* must not be less than value calculated with [NFTemplateGetSize](#) function.

The following flags are supported:

- [NFT_PROCESS_FIRST_RECORD_ONLY](#)
- Flags supported by [NFRecordSaveToMemory](#) function are applied to each finger record contained in the [NFTemplate](#).

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateGetSize](#) | [NFRecordSaveToMemory](#) | [NFTemplateCreateFromMemory](#)

6.5.3.19. NFTemplateSetRecordCapacity Function

Sets the number of finger records that the [NFTemplate](#) can contain.

```
NResult N_API NFTemplateSetRecordCapacity(
    HNFTemplate hTemplate,
    NInt value
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NFTemplate object.
<i>value</i>	[in] New number of finger records NFTemplate can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>value</i> is less than finger record count obtained using NFTemplateGetRecordCount function.

Remarks

Finger record capacity is the number of finger records that the [NFTemplate](#) can store. Finger record count (see [NFTemplateGetRecordCount](#) function) is the number of finger records that are actually in the [NFTemplate](#).

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NFTemplate Module](#) | [HNFTemplate](#) | [NFTemplateGetRecordCapacity](#) | [NFTemplateGetRecordCount](#)

6.5.3.20. NFTemplateUnpack Function

Unpacks packed finger records from the packed [NFTemplate](#).

```
NResult N_API NFTemplateUnpack(
```

```

    const void * buffer,
    NSizeType bufferSize,
    NByte * pMajorVersion,
    NByte * pMinorVersion,
    NUInt * pSize,
    NByte * pHeaderSize,
    NInt * pRecordCount,
    const void * * arRecords,
    NSizeType * arRecordSizes
);

```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NFTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NFTemplate.
<i>pMajorVersion</i>	[out] Pointer to NByte that receives major version of the packed NFTemplate. Can be NULL .
<i>pMinorVersion</i>	[out] Pointer to NByte that receives minor version of the packed NFTemplate. Can be NULL .
<i>pSize</i>	[out] Pointer to NUInt that receives size of the packed NFTemplate. Can be NULL .
<i>pHeaderSize</i>	[out] Pointer to NByte that receives header size of the packed NFTemplate. Can be NULL .
<i>pRecordCount</i>	[out] Pointer to NInt that receives number of finger records contained in the packed NFTemplate. Can be NULL .
<i>arRecords</i>	[out] Pointer to array of void * that receives pointers to packed finger records contained in the packed NFTemplate. Can be NULL .
<i>arRecordSizes</i>	[out] Pointer to array of NSizeType that receives sizes of packed finger records contained in the packed NFTemplate. Can be NULL .

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NFTemplate format.

Remarks

This is a low-level function and can be changed in future version of the library.

This function should be first called with *arRecords* and *arRecordSizes* parameters set to [NULL](#) to receive finger record count through *pRecordCount* parameter. Then arrays of finger records count length should be allocated and this function should be called with these array passed in *arRecords* and *arRecordSizes* parameters.

See Also

[NFTemplate Module](#) | [NFTemplatePack](#)

6.5.4. NLRecord Module

Provides functionality for packing, unpacking and editing Neurotechnology Face Records (NLRecords).

Header file: `NLRecord.h`

Functions

NLRecordCheck	Checks whether packed NLRecord is valid.
NLRecordClone	Creates a copy of the NLRecord.
NLRecordCreate	Creates an empty NLRecord.
NLRecordCreateFromMemory	Unpacks NLRecord from the specified memory buffer.
NLRecordFree	Deletes the NLRecord. After the object is deleted the specified handle is no longer valid.
NLRecordGetInfo	Retrieves library information into NLibraryInfo structure.
NLRecordGetMaxSize	Retrieves maximum possible size of serialized NLRecord in bytes.
NLRecordGetQuality	Retrieves the quality of the NLRecord.
NLRecordGetQualityMem	Retrieves the quality of the packed NLRecord.
NLRecordGetSize	Calculates size of serialized NLRecord.
<code>NLRecordInfoDispose</code>	For internal use.
NLRecordSaveToMemory	Serializes NLRecord into the specified memory buffer.
NLRecordSetQuality	Sets the quality of the NLRecord.

Structures

<code>NLRecordInfo</code>	For internal use.
---------------------------	-------------------

Types

<code>HNLRecord</code>	Handle to NLRecord object.
------------------------	----------------------------

See Also

[NTemplate Library](#)

6.5.4.1. NLRecordCheck Function

Checks whether packed NLRecord is valid.

```
NResult N_API NLRecordCheck(
    const void * buffer,
    NSizeType bufferSize
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLRecord.

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLRecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NLRecord Module](#)

6.5.4.2. NLRecordClone Function

Creates a copy of the NLRecord.

```
NResult N_API NLRecordClone(
    HNLRecord hRecord,
    HNLRecord * pHClonedRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NLRecord object.
<i>pHClonedRecord</i>	[out] Pointer to a HNLRecord that receives handle to newly created NLRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pHClonedRecord</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NLRecordFree](#) function.

See Also

[NLRecord Module](#) | [HNLRecord](#) | [NLRecordFree](#)

6.5.4.3. NLRecordCreate Function

Creates an empty NLRecord.

```
NResult N_API NLRecordCreate(
    NUInt flags,
    HNLRecord * pHRecord
);
```

Parameters

<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function. This parameter is reserved, must be zero.
<i>pHRecord</i>	[out] Pointer to HNLRecord that receives handle to created NLRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHRecord</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NLRecordFree](#) function.

See Also

[NLRecord Module](#) | [HNLRecord](#) | [NLRecordFree](#)

6.5.4.4. NLRecordCreateFromMemory Function

Unpacks NLRecord from the specified memory buffer.

```
NResult N_API NLRecordCreateFromMemory(
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NLRecordInfo * pInfo,
    HNLRecord * pHRecord
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLRecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pInfo</i>	[out] For internal use. Must be NULL .
<i>pHRecord</i>	[out] Pointer to HNLRecord that receives handle to newly created NLRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHRecord</i> or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLRecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NLRecordFree](#) function.

See Also

[NLRecord Module](#) | [HNLRecord](#) | [NLRecordInfo](#) | [NLRecordFree](#) | [NLRecordSaveToMemory](#)

6.5.4.5. NLRecordFree Function

Deletes the NLRecord. After the object is deleted the specified handle is no longer valid.

```
void N\_API NLRecordFree(
    HNLRecord hRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NLRecord object.
----------------	-------------------------------------

Remarks

If *hRecord* is [NULL](#), does nothing.

See Also

[NLRecord Module](#) | [HNLRecord](#)

6.5.4.6. NLRecordGetInfo Function

Retrieves library information into `NLibraryInfo` structure.

```
NResult N_API NRecordGetInfo(
    NLibraryInfo * pValue
);
```

Parameters

<i>pValue</i>	[out] Pointer to NLibraryInfo structure that receives library information.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NRecord Module](#) | [HNLRecord](#)

6.5.4.7. NRecordGetMaxSize Function

Retrieves maximum possible size of serialized `NRecord` in bytes.

```
NResult N_API NRecordGetMaxSize(
    NSizeType featuresSize,
    NSizeType * pSize
);
```

Parameters

<i>featuresSize</i>	[in] Size of internal features in bytes.
<i>pSize</i>	[out] Pointer to NSizeType that receives maximal size of packed <code>NRecord</code> .

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pSize</i> is NULL .

See Also

[NRecord Module](#) | [NRecordSaveToMemory](#)

6.5.4.8. NRecordGetQuality Function

Retrieves the quality of the NLRecord.

```
NResult N_API NLRecordGetQuality(
    HNLRecord hRecord,
    NByte * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NLRecord object.
<i>pValue</i>	[out] Pointer to NByte that receives face quality.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NLRecord Module](#) | [HNLRecord](#) | [NLRecordSetQuality](#)

6.5.4.9. NLRecordGetQualityMem Function

Retrieves the quality of the packed NLRecord.

```
NResult N_API NLRecordGetQualityMem(
    const void * buffer,
    NSizeType bufferSize,
    NByte * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLRecord.
<i>pValue</i>	[out] Pointer to NByte that receives face quality.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.

Error Code	Condition
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLRecord format.

Remarks

This function supports both NLRecord version 1.0 and 2.0 formats.

Always returns 0 for version 1.0 format.

See Also

[NLRecord Module](#)

6.5.4.10. NLRecordGetSize Function

Calculates size of serialized NLRecord.

```
NResult N_API NLRecordGetSize(
    HNLRecord hRecord,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NLRecord object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NLRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pSize</i> is NULL .

Remarks

For the list of flags that are supported see [NLRecordSaveToMemory](#) function.

See Also

[NLRecord Module](#) | [HNLRecord](#) | [NLRecordSaveToMemory](#)

6.5.4.11. NLRecordSaveToMemory Function

Serializes NLRecord into the specified memory buffer.

```
NResult N_API NLRecordSaveToMemory(
    HNLRecord hRecord,
    void * buffer,
```

```

    NSUInteger bufferSize,
    NSUInteger flags,
    NSUInteger * pSize
);

```

Parameters

<i>hRecord</i>	[in] Handle to the NLRecord object.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NLRecord. Can be NULL .
<i>bufferSize</i>	[in] Size of memory buffer to store packed NLRecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function. Should be zero.
<i>pSize</i>	[out] Pointer to NSUInteger that receives size of packed NLRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>buffer</i> is not NULL and <i>bufferSize</i> is less than size required to store packed NLRecord.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pSize</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is [NULL](#) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as [NLRecordGetSize](#) function.

If *buffer* is not [NULL](#), *bufferSize* must not be less than value calculated with [NLRecordGetSize](#) function.

See Also

[NLRecord Module](#) | [HNLRecord](#) | [NLRecordGetSize](#) | [NLRecordCreateFromMemory](#)

6.5.4.12. NLRecordSetQuality Function

Sets the quality of the NLRecord.

```

NResult N_API NLRecordSetQuality(
    HNLRecord hRecord,
    NByte value
);

```

Parameters

<i>hRecord</i>	[in] Handle to the NLRecord object.
----------------	-------------------------------------

<i>value</i>	[in] New face quality value.
--------------	------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NLRecord Module](#) | [HNLRecord](#) | [NLRecordGetQuality](#)

6.5.5. NLTemplate Module

Provides functionality for packing, unpacking and editing Neurotechnology Faces Templates (NLTemplates).

Header file: `NLTemplate.h`

Functions

NLTemplateAddRecord	Adds an empty face record to the NLTemplate.
NLTemplateAddRecordCopy	Adds a copy of the face record to the NLTemplate.
NLTemplateAddRecordFromMemory	Unpacks a face record from the specified memory buffer and adds it to the NLTemplate.
NLTemplateCalculateSize	Calculates the size of a packed NLTemplate containing the specified number of face records of specified size.
NLTemplateCheck	Checks if format of the packed NLTemplate is correct.
NLTemplateClearRecords	Removes all face records from the NLTemplate.
NLTemplateClone	Creates a copy of the NLTemplate.
NLTemplateCreate	Creates an empty NLTemplate.
NLTemplateCreateFromMemory	Unpacks a NLTemplate from the specified memory buffer.
NLTemplateFree	Deletes the NLTemplate. After the object is deleted the specified handle is no longer valid.
NLTemplateGetRecord	Retrieves the face record at the specified index of the NLTemplate.
NLTemplateGetRecordCapacity	Retrieves the number of face records that the NLTemplate can contain..
NLTemplateGetRecordCount	Retrieves the number of face records in the NLTemplate.
NLTemplateGetRecordCountMem	Retrieves the number of face records in the packed NLTemplate.
NLTemplateGetSize	Calculates packed size of the NLTemplate.
NLTemplateInfoDispose	For internal use.

NLTemplatePack	Packs packed face records as NLTemplate into the specified memory buffer.
NLTemplateRemoveRecord	Removes the face record at the specified index of the NLTemplate.
NLTemplateSaveToMemory	Packs the NLTemplate into the specified memory buffer.
NLTemplateSetRecordCapacity	Sets the number of face records that the NLTemplate can contain.
NLTemplateUnpack	Unpacks packed face records from the packed NLTemplate.

Structures

NLTemplateInfo	For internal use.
----------------	-------------------

Types

HNLTemplate	Handle to NLTemplate object.
-------------	------------------------------

Macros

NLT_MAX_RECORD_COUNT	The maximum number of face records NLTemplate can contain.
NLT_PROCESS_FIRST_RECORD_ONLY	The flag indicating whether only the first face record should be unpacked or packed while unpacking or packing NLTemplate.

See Also

[NTemplate Library](#)

6.5.5.1. NLTemplateAddRecord Function

Adds an empty face record to the NLTemplate.

```
NResult N_API NLTemplateAddRecord(
    HNLTemplate hTemplate,
    NUInt flags,
    HNLRecord * pHRecord
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pHRecord</i>	[out] Pointer to HNLRecord that receives handle to created NLRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHRecord</i> is NULL .
N_E_INVALID_OPERATION	Number of face records in NLTemplate (see NLTemplateGetRecordCount) is equal to NLT_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see [NLRecordCreate](#) function.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [HNLRecord](#) | [NLTemplateGetRecordCount](#) | [NLRecordCreate](#)

6.5.5.2. NLTemplateAddRecordCopy Function

Adds a copy of the face record to the NLTemplate.

```
NResult N_API NLTemplateAddRecordCopy(
    HNLTemplate hTemplate,
    HNLRecord hSrcRecord,
    HNLRecord * pHRecord
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
<i>hSrcRecord</i>	[in] Handle to the NLRecord object.
<i>pHRecord</i>	[out] Pointer to HNLRecord that receives handle to created NLRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>hSrcRecord</i> or <i>pHRecord</i> is NULL .
N_E_INVALID_OPERATION	Number of face records in NLTemplate (see NLTemplateGetRecordCount) is equal to NLT_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [HNLRecord](#) | [NLTemplateGetRecordCount](#)

6.5.5.3. NLTemplateAddRecordFromMemory Function

Unpacks a face record from the specified memory buffer and adds it to the NLTemplate.

```
NResult N_API NLTemplateAddRecordFromMemory(
    HNLTemplate hTemplate,
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    HNLRecord * pHRecord
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLRecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pHRecord</i>	[out] Pointer to HNLRecord that receives handle to created NLRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHRecord</i> , or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLRecord format.
N_E_INVALID_OPERATION	Number of face records in NLTemplate (see NLTemplateGetRecordCount) is equal to NLT_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see [NLRecordCreateFromMemory](#) function.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [HNLRecord](#) | [NLTemplateGetRecordCount](#) | [NLRecordCreateFromMemory](#)

6.5.5.4. NLTemplateCalculateSize Function

Calculates the size of a packed NLTemplate containing the specified number of face records of specified size.

```
NResult N_API NLTemplateCalculateSize(
    NInt recordCount,
    NSizeType * arRecordSizes,
    NSizeType * pSize
);
```

Parameters

<i>recordCount</i>	[in] The number of face records.
<i>arRecordSizes</i>	[in] Pointer to array of NSizeType that contains packed sizes of each face record.
<i>pSize</i>	[out] Pointer to NSizeType that receives calculated size of packed NLTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Any entry of the array <i>arRecordSizes</i> points to is less than minimal face record size.
N_E_ARGUMENT_NULL	<i>arRecordSizes</i> or <i>pSize</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>recordCount</i> is less than zero or greater than NLT_MAX_RECORD_COUNT .

Remarks

This is a low-level function and can be changed in future version of the library.

Use [NLRecordGetMaxSize](#) function to calculate packed size of an individual face record.

See Also

[NLTemplate Module](#) | [NLRecordGetMaxSize](#) | [NLTemplateSaveToMemory](#)

6.5.5.5. NLTemplateCheck Function

Checks if format of the packed NLTemplate is correct.

```
NResult N_API NLTemplateCheck(
    const void * buffer,
    NSizeType bufferSize
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLTemplate format.

See Also

[NLTemplate Module](#)

6.5.5.6. NLTemplateClearRecords Function

Removes all face records from the NLTemplate.

```
NResult N_API NLTemplateClearRecords(
    HNLTemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
------------------	---------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

See Also

[NLTemplate Module](#) | [HNLTemplate](#)

6.5.5.7. NLTemplateClone Function

Creates a copy of the NLTemplate.

```
NResult N_API NLTemplateClone(
    HNLTemplate hTemplate,
    HNLTemplate * pHClonedTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
<i>pHClonedTemplate</i>	[out] Pointer to HNLTemplate that receives handle to

	created NLTemplate object.
--	----------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHClonedTemplate</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NLTemplateFree](#) function.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateFree](#)

6.5.5.8. NLTemplateCreate Function

Creates an empty NLTemplate.

```
NResult N_API NLTemplateCreate(
    HNLTemplate * pHTemplate
);
```

Parameters

<i>pHTemplate</i>	[out] Pointer to HNLTemplate that receives handle to created NLTemplate object.
-------------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHTemplate</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NLTemplateFree](#) function.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateFree](#)

6.5.5.9. NLTemplateCreateFromMemory Function

Unpacks a NLTemplate from the specified memory buffer.

```

NResult N_API NLTemplateCreateFromMemory(
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NLTemplateInfo * pInfo,
    HNLTemplate * pHTemplate
);

```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLTemplate.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pInfo</i>	[out] For internal use. Must be NULL .
<i>pHTemplate</i>	[out] Pointer to HNLTemplate that receives handle to newly created NLTemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHTemplate</i> or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLTemplate format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- [NLT_PROCESS_FIRST_RECORD_ONLY](#)
- Flags supported by [NLRecordCreateFromMemory](#) function are applied to each face record contained in the NLTemplate.

Created object must be deleted using [NLTemplateFree](#) function.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateInfo](#) | [NLTemplateFree](#) | [NLRecordCreateFromMemory](#) | [NLTemplateSaveToMemory](#)

6.5.5.10. NLTemplateFree Function

Deletes the NLTemplate. After the object is deleted the specified handle is no longer valid.

```

void N_API NLTemplateFree(

```



```

    HNLTemplate hTemplate
);

```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
------------------	---------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

Remarks

If *hTemplate* is [NULL](#), does nothing.

See Also

[NLTemplate Module](#) | [HNLTemplate](#)

6.5.5.11. NLTemplateGetRecord Function

Retrieves the face record at the specified index of the NLTemplate.

```

NResult N_API NLTemplateGetRecord(
    HNLTemplate hTemplate,
    NInt index,
    HNLRecord * pValue
);

```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
<i>index</i>	[in] Index of face record to retrieve.
<i>pValue</i>	[out] Pointer to HNLRecord that receives handle to NLRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to face record count obtained using NLTemplateGetRecordCount function.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateGetRecordCount](#)

6.5.5.12. NLTemplateGetRecordCapacity Function

Retrieves the number of face records that the NLTemplate can contain..

```
NResult N_API NLTemplateGetRecordCapacity(
    HNLTemplate hTemplate,
    NInt * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
<i>pValue</i>	[out] Pointer to NInt that receives number of face records NLTemplate can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .

Remarks

Face record capacity is the number of face records that the NLTemplate can store. Face record count (see [NLTemplateGetRecordCount](#) function) is the number of face records that are actually in the NLTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding face records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateSetRecordCapacity](#) | [NLTemplateGetRecordCount](#)

6.5.5.13. NLTemplateGetRecordCount Function

Retrieves the number of face records in the NLTemplate.

```
NResult N_API NLTemplateGetRecordCount(
    HNLTemplate hTemplate,
    NInt * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to NLTemplate object.
<i>pValue</i>	[out] Pointer to NInt that receives number of face records.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .

Remarks

Face record capacity (see [NLTemplateGetRecordCapacity](#) and [NLTemplateSetRecordCapacity](#) functions) is the number of face records that the NLTemplate can store. Face record count is the number of face records that are actually in the NLTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding face records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateGetRecordCapacity](#) | [NLTemplateSetRecordCapacity](#)

6.5.5.14. NLTemplateGetRecordCountMem Function

Retrieves the number of face records in the packed NLTemplate.

```
NResult N_API NLTemplateGetRecordCountMem(
    const void * buffer,
    NSizeType bufferSize,
    NInt * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLTemplate.
<i>pValue</i>	[out] Pointer to NInt that receives number of face records.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLTemplate format.

See Also

[NLTemplate Module](#)

6.5.5.15. NLTemplateGetSize Function

Calculates packed size of the NLTemplate.

```

NResult N_API NLTemplateGetSize(
    HNLTemplate hTemplate,
    NUInt flags,
    NSizeType * pSize
);

```

Parameters

<i>hTemplate</i>	[in] Handle to NLTemplate object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives calculated size of packed NLTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pSize</i> is NULL .

Remarks

For the list of flags that are supported see [NLTemplateSaveToMemory](#) function.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateSaveToMemory](#)

6.5.5.16. NLTemplatePack Function

Packs packed face records as NLTemplate into the specified memory buffer.

```

NResult N_API NLTemplatePack(
    NInt recordCount,
    const void * * arRecords,
    NSizeType * arRecordSizes,
    void * buffer,
    NSizeType bufferSize,
    NSizeType * pSize
);

```

Parameters

<i>recordCount</i>	[in] The number of face records.
<i>arRecords</i>	[in] Pointer to array of void * that contains packed face

	records.
<i>arRecordSizes</i>	[in] Pointer to array of NSizeType that contains packed sizes of each face record.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NLTemplate.
<i>bufferSize</i>	[in] Size of memory buffer to store packed NLTemplate.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NLTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<p><i>bufferSize</i> is less than size required to store packed NLTemplate.</p> <p>- or -</p> <p>Any entry of the array <i>arRecordSizes</i> points to is less than minimal face record size.</p> <p>- or -</p> <p>Any entry of the array <i>arRecords</i> points to is NULL.</p> <p>- or -</p> <p>Any entry of the array <i>arRecordSizes</i> points to is not equal to size stored in memory buffer according entry of the array <i>arRecords</i> points to, points to.</p>
N_E_ARGUMENT_NULL	<i>arRecords</i> or <i>buffer</i> or <i>pSize</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>recordCount</i> is less than zero or greater than NLT_MAX_RECORD_COUNT .
N_E_FORMAT	Data in memory buffer any entry of the array <i>arRecords</i> points to is inconsistent with NLRecord format.

Remarks

This is a low-level function and can be changed in future version of the library.

bufferSize must not be less than value calculated with [NLTemplateCalculateSize](#) function with the same parameter values.

See Also

[NLTemplate Module](#) | [NLTemplateCalculateSize](#) | [NLTemplateUnpack](#)

6.5.5.17. NLTemplateRemoveRecord Function

Removes the face record at the specified index of the NLTemplate.

```
NResult N_API NLTemplateRemoveRecord(
    HNLTemplate hTemplate,
```

```

    NInt index
);

```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
<i>index</i>	[in] Index of face record to remove.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater or equal than records count.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateGetRecordCount](#)

6.5.5.18. NLTemplateSaveToMemory Function

Packs the NLTemplate into the specified memory buffer.

```

NResult N_API NLTemplateSaveToMemoryEx(
    HNLTemplate hTemplate,
    void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NSizeType * pSize
);

```

Parameters

<i>hTemplate</i>	[in] Handle to NLTemplate object.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NLTemplate. Can be NULL .
<i>bufferSize</i>	[in] Size of memory buffer to store packed NLTemplate.
<i>flags</i>	[in] Bitwise combination of zero or more flags that control behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NLTemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>buffer</i> is not NULL and <i>bufferSize</i> is less than size required to store packed NLTemplate.
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pSize</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is [NULL](#) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as [NLTemplateGetSize](#) function.

If *buffer* is not [NULL](#), *bufferSize* must not be less than value calculated with [NLTemplateGetSize](#) function.

The following flags are supported:

- [NLT_PROCESS_FIRST_RECORD_ONLY](#)
- Flags supported by [NLRecordSaveToMemory](#) function are applied to each face record contained in the NLTemplate.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateGetSize](#) | [NLRecordSaveToMemory](#) | [NLTemplateCreateFromMemory](#)

6.5.5.19. NLTemplateSetRecordCapacity Function

Sets the number of face records that the NLTemplate can contain.

```
NResult N_API NLTemplateSetRecordCapacity(
    HNLTemplate hTemplate,
    NInt value
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NLTemplate object.
<i>value</i>	[in] New number of face records NLTemplate can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>value</i> is less than face record count obtained using NLTemplateGetRecordCount function.

Remarks

Face record capacity is the number of face records that the NLTemplate can store. Face record count (see [NLTemplateGetRecordCount](#) function) is the number of face records that are actually in the NLTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding face records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NLTemplate Module](#) | [HNLTemplate](#) | [NLTemplateGetRecordCapacity](#) | [NLTemplateGetRecordCount](#)

6.5.5.20. NLTemplateUnpack Function

Unpacks packed face records from the packed NLTemplate.

```
NResult N_API NLTemplateUnpack(
    const void * buffer,
    NSizeType bufferSize,
    NByte * pMajorVersion,
    NByte * pMinorVersion,
    NUInt * pSize,
    NByte * pHeaderSize,
    NInt * pRecordCount,
    const void * * arRecords,
    NSizeType * arRecordSizes
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NLTemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NLTemplate.
<i>pMajorVersion</i>	[out] Pointer to NByte that receives major version of the packed NLTemplate. Can be NULL .
<i>pMinorVersion</i>	[out] Pointer to NByte that receives minor version of the packed NLTemplate. Can be NULL .
<i>pSize</i>	[out] Pointer to NUInt that receives size of the packed NLTemplate. Can be NULL .
<i>pHeaderSize</i>	[out] Pointer to NByte that receives header size of the packed NLTemplate. Can be NULL .
<i>pRecordCount</i>	[out] Pointer to NInt that receives number of face records contained in the packed NLTemplate. Can be NULL .
<i>arRecords</i>	[out] Pointer to array of void * that receives pointers to packed face records contained in the packed NLTemplate. Can be NULL .
<i>arRecordSizes</i>	[out] Pointer to array of NSizeType that receives sizes of packed face records contained in the packed NLTemplate. Can be NULL .

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NLTemplate format.

Remarks

This is a low-level function and can be changed in future version of the library.

This function should be first called with *arRecords* and *arRecordSizes* parameters set to [NULL](#) to receive face record count through *pRecordCount* parameter. Then arrays of face records count length should be allocated and this function should be called with these array passed in *arRecords* and *arRecordSizes* parameters.

See Also

[NLTemplate Module](#) | [NLTemplatePack](#)

6.5.6. NETemplate Module

Provides functionality for packing, unpacking and editing Neurotechnology iris templates (NETemplates).

Header file: `NETemplate.h`

Functions

NETemplateCalculateSize	Calculates the size of a packed NETemplate containing the specified number of iris records of specified size.
NETemplatePack	Packs packed iris records as a NETemplate into the specified memory buffer.
NETemplateUnpack	Unpacks packed iris records from the packed NETemplate.
NETemplateCheck	Checks if format of the packed NETemplate is correct.
NETemplateGetRecordCountMem	Retrieves the number of iris records in the packed NETemplate.
NETemplateCreate	Creates an empty NETemplate.
NETemplateCreate	Creates an empty NETemplate.
NETemplateCreateFromMemory	Unpacks a NETemplate from the specified memory buffer.
NETemplateFree	Frees memory taken up by NETemplate object.
NETemplateGetRecordCount	Retrieves the number of iris record in the NETemplate.
NETemplateGetRecord	Retrieves the iris record at the specified index of the NLTemplate.
NETemplateGetRecordCapacity	Retrieves the maximum number of iris records that the NLTemplate can contain.
NETemplateSetRecordCapacity	Sets the maximum number of iris records that the NLTemplate can contain.

NETemplateAddRecord	Adds an empty iris record to the NETemplate.
NETemplateAddRecordFromMemory	Unpacks an iris record from the specified memory buffer and adds it to the NETemplate.
NETemplateAddRecordCopy	Adds a copy of the iris record to the NETemplate.
NETemplateRemoveRecord	Removes the iris record at the specified index of the NETemplate.
NETemplateClearRecords	Removes all iris records from the NETemplate.
NETemplateClone	Creates a copy of the NETemplate.
NETemplateGetSize	Calculates packed size of the NETemplate.
NETemplateSaveToMemory	Packs the NETemplate into the specified memory buffer.

Structures

NETemplateInfo	For internal use.
--------------------------------	-------------------

Types

HNETemplate	Handle to NETemplate object.
-----------------------------	------------------------------

Macros

NET_MAX_RECORD_COUNT	The maximum number of iris records NETemplate can contain.
NLT_PROCESS_FIRST_RECORD_ONLY	The flag indicating whether only the first iris record should be unpacked or packed while unpacking or packing NETemplate.

See Also

[NETemplate Library](#)

6.5.6.1. NETemplateCalculateSize Function

Calculates the size of a packed NETemplate containing the specified number of iris records of specified size.

```
NResult N_API NETemplateCalculateSize(
    NInt recordCount,
    NSizeType * arRecordSizes,
    NSizeType * pSize
);
```

Parameters

<i>recordCount</i>	[in] The number of iris records.
<i>arRecordSizes</i>	[in] Pointer to array of NSizeType that contains packed sizes of each iris record.
<i>pSize</i>	[out] Pointer to NSizeType that receives calculated size

	of packed NETemplate.
--	-----------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Any entry of the array <i>arRecordSizes</i> points to is less than minimal face record size.
N_E_ARGUMENT_NULL	<i>arRecordSizes</i> or <i>pSize</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>recordCount</i> is less than zero or greater than NET_MAX_RECORD_COUNT .

Remarks

This is a low-level function and can be changed in future version of the library.

Use [NERecordGetMaxSize](#) function to calculate packed size of an individual iris record.

See Also

[NETemplate Module](#) | [NERecordGetMaxSize](#) | [NETemplateSaveToMemory](#)

6.5.6.2. NETemplatePack Function

Packs packed iris records as a NETemplate into the specified memory buffer.

```
NResult N_API NETemplatePack(
    NInt recordCount,
    const void * * arRecords,
    NSizeType * arRecordSizes,
    void * buffer,
    NSizeType bufferSize,
    NSizeType * pSize
);
```

Parameters

<i>recordCount</i>	[in] The number of iris records.
<i>arRecords</i>	[in] Pointer to array of void * that contains packed iris records.
<i>arRecordSizes</i>	[in] Pointer to array of NSizeType that contains packed sizes of each iris record.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NETemplate.
<i>bufferSize</i>	[in] Size of memory buffer to store packed NETemplate.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NETemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<p><i>bufferSize</i> is less than size required to store packed NETemplate.</p> <p>- or -</p> <p>Any entry of the array <i>arRecordSizes</i> points to is less than minimal iris record size.</p> <p>- or -</p> <p>Any entry of the array <i>arRecords</i> points to is NULL.</p> <p>- or -</p> <p>Any entry of the array <i>arRecordSizes</i> points to is not equal to size stored in memory buffer according entry of the array <i>arRecords</i> points to, points to.</p>
N_E_ARGUMENT_NULL	<i>arRecords</i> or <i>buffer</i> or <i>pSize</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>recordCount</i> is less than zero or greater than NET_MAX_RECORD_COUNT .
N_E_FORMAT	Data in memory buffer any entry of the array <i>arRecords</i> points to is inconsistent with NERecord format.

Remarks

This is a low-level function and can be changed in future version of the library.

bufferSize must not be less than value calculated with [NETemplateCalculateSize](#) function with the same parameter values.

See Also

[NETemplate Module](#) | [NETemplateCalculateSize](#) | [NETemplateUnpack](#)

6.5.6.3. NETemplateUnpack Function

Unpacks packed iris records from the packed NETemplate.

```
NResult N_API NETemplateUnpack(
    const void * buffer,
    NSizeType bufferSize,
    NByte * pMajorVersion,
    NByte * pMinorVersion,
    NUInt * pSize,
    NByte * pHeaderSize,
    NInt * pRecordCount,
    const void * * arRecords,
    NSizeType * arRecordSizes
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NETemplate.
---------------	--

<i>bufferSize</i>	[in] Size of memory buffer that contains packed NETemplate.
<i>pMajorVersion</i>	[out] Pointer to NByte that receives major version of the packed NETemplate. Can be NULL .
<i>pMinorVersion</i>	[out] Pointer to NByte that receives minor version of the packed NETemplate. Can be NULL .
<i>pSize</i>	[out] Pointer to NUInt that receives size of the packed NETemplate. Can be NULL .
<i>pHeaderSize</i>	[out] Pointer to NByte that receives header size of the packed NETemplate. Can be NULL .
<i>pRecordCount</i>	[out] Pointer to NInt that receives number of iris records contained in the packed NETemplate. Can be NULL .
<i>arRecords</i>	[out] Pointer to array of void * that receives pointers to packed iris records contained in the packed NETemplate. Can be NULL .
<i>arRecordSizes</i>	[out] Pointer to array of NSizeType that receives sizes of packed iris records contained in the packed NETemplate. Can be NULL .

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NETemplate format.

Remarks

This is a low-level function and can be changed in future version of the library.

This function should be first called with *arRecords* and *arRecordSizes* parameters set to [NULL](#) to receive iris record count through *pRecordCount* parameter. Then arrays of iris records count length should be allocated and this function should be called with these array passed in *arRecords* and *arRecordSizes* parameters.

See Also

[NETemplate Module](#) | [NETemplatePack](#)

6.5.6.4. NETemplateCheck Function

Checks if format of the packed NETemplate is correct.

```
NResult N_API NETemplateCheck(
    const void * buffer,
    NSizeType bufferSize
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NETemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NETemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NETemplate format.

See Also

[NETemplate Module](#)

6.5.6.5. NETemplateGetRecordCountMem Function

Retrieves the number of iris records in the packed NETemplate.

```
NResult N_API NETemplateGetRecordCountMem(
    const void * buffer,
    NSizeType bufferSize,
    NInt * pValue
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NETemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NETemplate.
<i>pValue</i>	[out] Pointer to NInt that receives number of iris records.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent

Error Code	Condition
	with NETemplate format.

See Also

[NETemplate Module](#)

6.5.6.6. NETemplateInfoDispose Function

Frees memory taken up by NETemplateInfo structure.

```
void N_API NETemplateInfoDispose(NERecordInfo * pInfo);
```

Parameters

<i>pValue</i>	[in] Pointer to NERecordInfo structure that holds NERecord info.
---------------	--

See Also

[NETemplate Module](#)

6.5.6.7. NETemplateCreate Function

Creates an empty NETemplate.

```
NResult N_API NETemplateCreate(
    HNETemplate * pHTemplate
);
```

Parameters

<i>pHTemplate</i>	[out] Pointer to HNETemplate that receives handle to created NETemplate object.
-------------------	---

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHTemplate</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NETemplateFree](#) function.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateFree](#)

6.5.6.8. NETemplateCreateFromMemory Function

Unpacks a NETemplate from the specified memory buffer.

```
NResult N_API NETemplateCreateFromMemory(
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NETemplateInfo * pInfo,
    HNETemplate * pHTemplate
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NETemplate.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NETemplate.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pInfo</i>	[out] For internal use. Must be NULL .
<i>pHTemplate</i>	[out] Pointer to HNETemplate that receives handle to newly created NETemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHTemplate</i> or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NETemplate format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- [NLT_PROCESS_FIRST_RECORD_ONLY](#)
- Flags supported by [NERecordCreateFromMemory](#) function are applied to each iris record contained in the NETemplate.

Created object must be deleted using [NETemplateFree](#) function.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateInfo](#) | [NETemplateFree](#) | [NERecordCreateFromMemory](#) | [NETemplateSaveToMemory](#)

6.5.6.9. NETemplateFree Function

Frees memory taken up by NETemplate object.

```
void N_API NETemplateFree(
    HNETemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
------------------	---------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

Remarks

If *hTemplate* is [NULL](#), does nothing.

See Also

[NETemplate Module](#) | [HNETemplate](#)

6.5.6.10. NETemplateGetRecordCount Function

Retrieves the number of iris record in the NETemplate.

```
NResult N_API NETemplateGetRecordCount(
    HNETemplate hTemplate,
    NInt * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to NETemplate object.
<i>pValue</i>	[out] Pointer to NInt that receives number of iris records.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .

Remarks

iris record capacity (see [NETemplateGetRecordCapacity](#) and [NETemplateSetRecordCapacity](#) functions) is the number of iris records that the NETemplate can store. iris record count is the number of iris records that are

actually in the NETemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding iris records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateGetRecordCapacity](#) | [NETemplateSetRecordCapacity](#)

6.5.6.11. NETemplateGetRecord Function

Retrieves the iris record at the specified index of the NLTemplate.

```
NResult N_API NETemplateGetRecord(
    HNETemplate hTemplate,
    NInt index,
    HNERecord * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
<i>index</i>	[in] Index of iris record to retrieve.
<i>pValue</i>	[out] Pointer to HNERecord that receives handle to NERecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater than or equal to iris record count obtained using NETemplateGetRecordCount function.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateGetRecordCount](#)

6.5.6.12. NETemplateGetRecordCapacity Function

Retrieves the maximum number of iris records that the NLTemplate can contain.

```
NResult N_API NETemplateGetRecordCapacity(
    HNETemplate hTemplate,
    NInt * pValue
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
<i>pValue</i>	[out] Pointer to NInt that receives number of iris records

	NETemplate can contain.
--	-------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pValue</i> is NULL .

Remarks

iris record capacity is the number of iris records that the NETemplate can store. iris record count (see [NETemplate-GetRecordCount](#) function) is the number of iris records that are actually in the NETemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding iris records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateSetRecordCapacity](#) | [NETemplateGetRecordCount](#)

6.5.6.13. NETemplateSetRecordCapacity Function

Sets the maximum number of iris records that the NLTemplate can contain.

```
NResult N_API NETemplateSetRecordCapacity(
    HNETemplate hTemplate,
    NInt value
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
<i>value</i>	[in] New number of iris records NETemplate can contain.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>value</i> is less than iris record count obtained using NETemplateGetRecordCount function.

Remarks

iris record capacity is the number of iris records that the NETemplate can store. iris record count (see [NETemplate-GetRecordCount](#) function) is the number of iris records that are actually in the NETemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding iris records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateGetRecordCapacity](#) | [NETemplateGetRecordCount](#)

6.5.6.14. NETemplateAddRecord Function

Adds an empty iris record to the NLTemplate.

```
NResult N_API NETemplateAddRecord(
    HNETemplate hTemplate,
    NUInt flags,
    HNERecord * pHRecord
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pHRecord</i>	[out] Pointer to HNERecord that receives handle to created NERecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHRecord</i> is NULL .
N_E_INVALID_OPERATION	Number of iris records in NETemplate (see NETemplateGetRecordCount) is equal to NET_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see [NERecordCreate](#) function.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [HNERecord](#) | [NETemplateGetRecordCount](#) | [NERecordCreate](#)

6.5.6.15. NETemplateAddRecordFromMemory Function

Unpacks an iris record from the specified memory buffer and adds it to the NETemplate.

```
NResult N_API NETemplateAddRecordFromMemory(
    HNETemplate hTemplate,
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    HNERecord * pHRecord
);
```

```
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
<i>buffer</i>	[in] Pointer to memory buffer that contains packed NERecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NERecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pHRecord</i>	[out] Pointer to HNERecord that receives handle to created NERecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHRecord</i> , or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NERecord format.
N_E_INVALID_OPERATION	Number of iris records in NETemplate (see NETemplateGetRecordCount) is equal to NET_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see [NERecordCreateFromMemory](#) function.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [HNERecord](#) | [NETemplateGetRecordCount](#) | [NERecordCreateFromMemory](#)

6.5.6.16. NETemplateAddRecordCopy Function

Adds a copy of the iris record to the NETemplate.

```
NResult N_API NETemplateAddRecordCopy(
    HNETemplate hTemplate,
    HNERecord hSrcRecord,
    HNERecord * pHRecord
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
------------------	---------------------------------------

<i>hSrcRecord</i>	[in] Handle to the NERecord object.
<i>pHRecord</i>	[out] Pointer to HNERecord that receives handle to created NERecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>hSrcRecord</i> or <i>pHRecord</i> is NULL .
N_E_INVALID_OPERATION	Number of iris records in NETemplate (see NETemplateGetRecordCount) is equal to NET_MAX_RECORD_COUNT .
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [HNERecord](#) | [NETemplateGetRecordCount](#)

6.5.6.17. NETemplateRemoveRecord Function

Removes the iris record at the specified index of the NETemplate.

```
NResult N_API NETemplateRemoveRecord(
    HNETemplate hTemplate,
    NInt index
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
<i>index</i>	[in] Index of iris record to remove.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .
N_E_ARGUMENT_OUT_OF_RANGE	<i>index</i> is less than zero or greater or equal than records count.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateGetRecordCount](#)

6.5.6.18. NETemplateClearRecords Function

Removes all iris records from the NETemplate.

```
NResult N_API NETemplateClearRecords(
    HNETemplate hTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
------------------	---------------------------------------

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> is NULL .

See Also

[NETemplate Module](#) | [HNETemplate](#)

6.5.6.19. NETemplateClone Function

Creates a copy of the NETemplate.

```
NResult N_API NETemplateClone(
    HNETemplate hTemplate,
    HNETemplate * pHClonedTemplate
);
```

Parameters

<i>hTemplate</i>	[in] Handle to the NETemplate object.
<i>pHClonedTemplate</i>	[out] Pointer to HNETemplate that receives handle to created NETemplate object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pHClonedTemplate</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NETemplateFree](#) function.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateFree](#)

6.5.6.20. NETemplateGetSize Function

Calculates packed size of the NETemplate.

```
NResult N_API NETemplateGetSize(
    HNETemplate hTemplate,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hTemplate</i>	[in] Handle to NETemplate object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives calculated size of packed NETemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pSize</i> is NULL .

Remarks

For the list of flags that are supported see [NETemplateSaveToMemory](#) function.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateSaveToMemory](#)

6.5.6.21. NETemplateSaveToMemory Function

Packs the NETemplate into the specified memory buffer.

```
NResult N_API NETemplateSaveToMemoryEx(
    HNETemplate hTemplate,
    void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hTemplate</i>	[in] Handle to NETemplate object.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NETemplate. Can be NULL .
<i>bufferSize</i>	[in] Size of memory buffer to store packed NETemplate.

<i>flags</i>	[in] Bitwise combination of zero or more flags that control behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NETemplate.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>buffer</i> is not NULL and <i>bufferSize</i> is less than size required to store packed NETemplate.
N_E_ARGUMENT_NULL	<i>hTemplate</i> or <i>pSize</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is [NULL](#) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as [NETemplateGetSize](#) function.

If *buffer* is not [NULL](#), *bufferSize* must not be less than value calculated with [NETemplateGetSize](#) function.

The following flags are supported:

- [NET_PROCESS_FIRST_RECORD_ONLY](#)
- Flags supported by [NERecordSaveToMemory](#) function are applied to each iris record contained in the NETemplate.

See Also

[NETemplate Module](#) | [HNETemplate](#) | [NETemplateGetSize](#) | [NERecordSaveToMemory](#) | [NETemplateCreateFromMemory](#)

6.5.7. NERecord Module

Provides functionality for packing, unpacking and editing Neurotechnology iris Records (NERecords).

Header file: `NERecord.h`

Functions

NERecordCheck	Checks whether packed NERecord is valid.
NERecordClone	Creates a copy of the NERecord.
NERecordCreate	Creates an empty NERecord.
NERecordCreateFromMemory	Unpacks NERecord from the specified memory buffer.
NERecordFree	Deletes the NERecord. After the object is deleted the specified handle is no longer valid.
NERecordGetInfo	Retrieves library information into NLibraryInfo struc-

	ture.
NERecordGetMaxSize	Retrieves maximum possible size of serialized NERecord in bytes.
NERecordGetQuality	Retrieves the quality of the NERecord.
NERecordGetQualityMem	Retrieves the quality of the packed NERecord.
NERecordGetSize	Calculates size of serialized NERecord.
NERecordInfoDispose	Frees memory taken up by NERecordInfo structure.
NERecordSaveToMemory	Serializes NERecord into the specified memory buffer.
NERecordSetQuality	Sets the quality of the NERecord.
NERecordGetWidthMem	Retrieves the image width of the packed NERecord.
NERecordGetHeightMem	Retrieves the image height of the packed NERecord.
NERecordGetPositionMem	Retrieves the image height of the packed NERecord.

Structures

NERecordInfo	For internal use.
--------------	-------------------

Enumerations

NEPosition	Enumerates eye position values.
------------	---------------------------------

Types

HNERecord	Handle to NERecord object.
-----------	----------------------------

See Also

[NTemplate Library](#)

6.5.7.1. NEPosition enumeration

Enumerates eye position values.

```
typedef enum NEPosition_
{
    nepUnknown = 0,
    nepRight = 1,
    nepLeft = 2,
} NEPosition;
```

Members

nepUnknown	Unknown eye.
nepRight	Right eye.
nepLeft	Left eye.

See Also

[NRecord Module](#).

6.5.7.2. NRecordCheck Function

Checks whether packed NRecord is valid.

```
NResult N_API NRecordCheck(
    const void * buffer,
    NSizeType bufferSize
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NRecord.

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NRecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

[NRecord Module](#)

6.5.7.3. NRecordClone Function

Creates a copy of the NRecord.

```
NResult N_API NRecordClone(
    HNERecord hRecord,
    HNERecord * pHClonedRecord
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NRecord object.
<i>pHClonedRecord</i>	[out] Pointer to a HNERecord that receives handle to newly created NRecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pHClonedRecord</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NERecordFree](#) function.

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordFree](#)

6.5.7.4. NERecordCreate Function

Creates an empty NERecord.

```
NResult N_API NERecordCreate(
    NUInt flags,
    HNERecord * pHRecord
);
```

Parameters

<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function. This parameter is reserved, must be zero.
<i>pHRecord</i>	[out] Pointer to HNERecord that receives handle to created NERecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHRecord</i> is NULL .
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NERecordFree](#) function.

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordFree](#)

6.5.7.5. NERecordCreateFromMemory Function

Unpacks NERecord from the specified memory buffer.

```

NResult N_API NERecordCreateFromMemory(
    const void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NERecordInfo * pInfo,
    HNERecord * pHRecord
);

```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NERecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NERecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pInfo</i>	[out] For internal use. Must be NULL .
<i>pHRecord</i>	[out] Pointer to HNERecord that receives handle to newly created NERecord object.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pHRecord</i> or <i>buffer</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NERecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using [NERecordFree](#) function.

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordInfo](#) | [NERecordFree](#) | [NERecordSaveToMemory](#)

6.5.7.6. NERecordFree Function

Deletes the NERecord. After the object is deleted the specified handle is no longer valid.

```

void N_API NERecordFree(
    HNERecord hRecord
);

```

Parameters

<i>hRecord</i>	[in] Handle to the NERecord object.
----------------	-------------------------------------

Remarks

If *hRecord* is [NULL](#), does nothing.

See Also

[NRecord Module](#) | [HNERRecord](#)

6.5.7.7. NRecordGetInfo Function

Retrieves library information into NLibraryInfo structure.

```
NResult N_API NRecordGetInfo(
    NLibraryInfo * pValue
);
```

Parameters

<i>pValue</i>	[out] Pointer to NLibraryInfo structure that receives library information.
---------------	--

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NRecord Module](#) | [HNERRecord](#)

6.5.7.8. NRecordInfoDispose Function

Frees memory taken up by NRecordInfo structure.

```
void N_API NRecordInfoDispose(NRecordInfo * pInfo);
```

Parameters

<i>pValue</i>	[in] Pointer to NRecordInfo structure that holds NRecord info.
---------------	--

See Also

[NRecord Module](#)

6.5.7.9. NRecordGetMaxSize Function

Retrieves maximum possible size of serialized NRecord in bytes.

```
NResult N_API NRecordGetMaxSize(
    NSizeType featuresSize,
    NSizeType * pSize
);
```

Parameters

<i>featuresSize</i>	[in] Size of internal features in bytes.
<i>pSize</i>	[out] Pointer to NSizeType that receives maximal size of packed NERecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pSize</i> is NULL .

See Also

[NERecord Module](#) | [NERecordSaveToMemory](#)

6.5.7.10. NERecordGetQuality Function

Retrieves the quality of the NERecord.

```
NResult N_API NERecordGetQuality(
    HNERecord hRecord,
    NByte * pValue
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NERecord object.
<i>pValue</i>	[out] Pointer to NByte that receives iris quality.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pValue</i> is NULL .

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordSetQuality](#)

6.5.7.11. NERecordGetQualityMem Function

Retrieves the quality of the packed NERecord.

```
NResult N_API NERecordGetQualityMem(
    const void * buffer,
    NSizeType bufferSize,
    NByte * pValue
);
```

```
);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NERRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NERRecord.
<i>pValue</i>	[out] Pointer to NByte that receives iris quality.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .
N_E_END_OF_STREAM	<i>bufferSize</i> is less than expected.
N_E_FORMAT	Data in memory buffer <i>buffer</i> points to is inconsistent with NERRecord format.

Remarks

This function supports both NERRecord version 1.0 and 2.0 formats.

Always returns 0 for version 1.0 format.

See Also

[NERRecord Module](#)

6.5.7.12. NERRecordGetSize Function

Calculates size of serialized NERRecord.

```
NResult N_API NERRecordGetSize(
    HNERRecord hRecord,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NERRecord object.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NERRecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pSize</i> is NULL .

Remarks

For the list of flags that are supported see [NERecordSaveToMemory](#) function.

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordSaveToMemory](#)

6.5.7.13. NERecordSaveToMemory Function

Serializes NERecord into the specified memory buffer.

```
NResult N_API NERecordSaveToMemory(
    HNERecord hRecord,
    void * buffer,
    NSizeType bufferSize,
    NUInt flags,
    NSizeType * pSize
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NERecord object.
<i>buffer</i>	[out] Pointer to memory buffer to store packed NERecord. Can be NULL .
<i>bufferSize</i>	[in] Size of memory buffer to store packed NERecord.
<i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function. Should be zero.
<i>pSize</i>	[out] Pointer to NSizeType that receives size of packed NERecord.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	<i>buffer</i> is not NULL and <i>bufferSize</i> is less than size required to store packed NERecord.
N_E_ARGUMENT_NULL	<i>hRecord</i> or <i>pSize</i> is NULL . - or - <i>buffer</i> is NULL and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is [NULL](#) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as [NERRecordGetSize](#) function.

If *buffer* is not [NULL](#), *bufferSize* must not be less than value calculated with [NERRecordGetSize](#) function.

See Also

[NERRecord Module](#) | [HNERRecord](#) | [NERRecordGetSize](#) | [NERRecordCreateFromMemory](#)

6.5.7.14. NERRecordSetQuality Function

Sets the quality of the NERRecord.

```
NResult N_API NERRecordSetQuality(
    HNERRecord hRecord,
    NByte value
);
```

Parameters

<i>hRecord</i>	[in] Handle to the NERRecord object.
<i>value</i>	[in] New iris quality value.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>hRecord</i> is NULL .

See Also

[NERRecord Module](#) | [HNERRecord](#) | [NERRecordGetQuality](#)

6.5.7.15. NERRecordGetWidthMem Function

Retrieves the image width of the packed NERRecord.

```
NResult N_API NERRecordGetWidthMem(const void *buffer, NSizeType bufferSize, NUShort *pValue);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NERRecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NERRecord.
<i>pValue</i>	[out] Pointer to NUShort that receives width of iris image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordGetHeightMem](#)

6.5.7.16. NERecordGetHeightMem Function

Retrieves the image height of the packed NERecord.

```
NResult N_API NERecordGetHeightMem(const void *buffer, NSizeType bufferSize, NUShort *pValue);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NERecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NERecord.
<i>pValue</i>	[out] Pointer to NUShort that receives height of iris image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordGetWidthMem](#)

6.5.7.17. NERecordGetPositionMem Function

Retrieves the image height of the packed NERecord.

```
NResult N_API NERecordGetPositionMem(const void *buffer, NSizeType bufferSize, NEPosition *pValue);
```

Parameters

<i>buffer</i>	[in] Pointer to memory buffer that contains packed NERecord.
<i>bufferSize</i>	[in] Size of memory buffer that contains packed NERecord.
<i>pValue</i>	[out] Pointer to NEPosition that receives the position of iris image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>buffer</i> or <i>pValue</i> is NULL .

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordGetPosition](#) | [NERecordSetPosition](#)

6.5.7.18. NERecordGetWidth Function

Retrieves the image width of the NERecord.

```
NResult N_API NERecordGetWidth(HNERecord hRecord, NUShort *pValue);
```

Parameters

<i>hRecord</i>	[in] handle to a NERecord.
<i>pValue</i>	[out] Pointer to NUShort that receives width of an iris image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordGetHeightMem](#)

6.5.7.19. NERecordGetHeight Function

Retrieves the image height of the NERecord.

```
NResult N_API NERecordGetHeight(HNERecord hRecord, NUShort *pValue);
```

Parameters

<i>hRecord</i>	[in] handle to a NERecord.
<i>pValue</i>	[out] Pointer to NUShort that receives height of an iris image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordGetHeightMem](#)

6.5.7.20. NERecordGetPosition Function

Gets the eye position of the NERecord.

```
NResult N_API NERecordGetPosition(HNERecord hRecord, NEPosition *pValue);
```

Parameters

<i>hRecord</i>	[in] handle to a NERecord.
<i>pValue</i>	[out] Pointer to NEPosition that receives the position of iris image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NERecord Module](#) | [HNERecord](#) | [NERecordGetPositionMem](#) | [NERecordSetPosition](#)

6.5.7.21. NERecordSetPosition Function

Sets the eye position of the NERecord.

```
NResult N_API NERecordSetPosition(HNERecord hRecord, NEPosition value);
```

Parameters

<i>hRecord</i>	[in] handle to a NERecord.
<i>pValue</i>	[in] NEPosition type variable that sets the position property of an iris image.

Return Values

If the function succeeds, the return value is [N_OK](#).

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	<i>pValue</i> is NULL .

See Also

[NRecord Module](#) | [HNERecord](#) | [NRecordGetPositionMem](#) | [NRecordSetPosition](#)

Appendix A. Support

Neurotechnology provides customer support during the entire period, while the customer develops and uses his own system based on our products. Customers are welcome to contact:

- <support@neurotechnology.com> for any help on solving based on FaceCell EDK development problems.

Appendix B. Distribution Content

FaceCell EDK distribution contains the following folders and files:

B.1. bin

Subdirectories of this directory contain shared libraries ,binary files of demo applications, and activation files for corresponding operating systems.

linux_arm	Contains FaceCell EDK binary files for linux OS (platform arm).
ppc03_armv4	Contains FaceCell EDK binary files for MS Windows CE 2003 OS (platform arm).
ppc03_ipaq	Contains FaceCell EDK binary files for MS Windows CE 2003 OS (platform arm) or later for users using ipaq 5500 series devices.
ppc05_armv4i	Contains FaceCell EDK binary files for MS Windows CE 2005 OS (platform arm).

B.2. documentation

Contains comprehensive documentation of the FaceCell EDK and the license.

B.3. include

Subdirectories of this directory contains include files for corresponding operating systems.

linux	Contains FaceCell EDK include files for linux OS (platform arm).
Windows	Contains FaceCell EDK include files for MS Windows OS (platform arm).

B.4. lib

Subdirectories of this directory contains library files for corresponding operating systems.

linux	Contains FaceCell EDK library files for linux OS (platform arm) .
ppc03_armv4	Contains FaceCell EDK library files for MS Windows CE 2003 (platform arm) .
ppc05_armv4i	Contains FaceCell EDK library files for MS Windows CE 2005 (platform arm).

B.5. samples

Subdirectories of this directory contains source code of demo applications for corresponding platforms.

Windows	Contains C source code of demo applications for Windows.
---------	--

B.6. tutorial

Subdirectories of this directory contains tutorial programs source code for different programming languages.

C	Contains C source code of tutorial applications.
---	--

Appendix C. Change Log

This appendix lists FaceCell EDK components changes among versions.

Legend

- FIX - bug was fixed.
- CHN - some changes were made.
- UPD - something has been updated.
- ADD - something has been added.
- REM - something has been removed.

Version 1.2.0.0

- UPD: FccExtractor library to version [1.2.0.0](#).
- UPD: NCore library to version [2.4.4.0](#).
- UPD: NImages library to version [2.4.0.0](#).
- CHN: NLRecord library replaced with NTemplate library version [1.4.0.1](#).
- UPD: Windows FccSample.cpp to version [1.1.0.0](#).

Version 1.1.0.0

- UPD: FccExtractor library to version [1.1.0.0](#).
- UPD: FccMatcher library to version [1.1.0.0](#).
- UPD: NCore library to version [2.4.1.0](#).
- UPD: NImages library to version [2.2.0.2](#).
- UPD: Windows FccSample.cpp to version [1.0.1.0](#).

Version 1.0.1.0

- UPD: FccExtractor library to version [1.0.1.0](#).
- UPD: NCore library to version [2.4.0.0](#).
- UPD: NImages library to version [2.2.0.1](#).

Version 1.0.0.2

- UPD: NImages library to version [2.1.0.2](#).
- UPD: FccExtractor library to version [1.0.0.1](#).

Version 1.0.0.1

- UPD: NImages library to version [2.1.0.0](#).
- UPD: Windows FccSample.cpp to version [1.0.0.2](#).

Version 1.0.0.0

- Initial release.

C.1. Components

C.1.1. FccExtractor Library

Version 1.2.0.0

- UPD: Improved face detection.

Version 1.1.0.0

- UPD: Improved face and eyes detection.
- UPD: Improved enrollment from sequence of images (generates more reliable templates).
- ADD: FCCEP_FACE_QUALITY_THRESHOLD parameter to control quality of face images.

Version 1.0.1.1

- ADD: FcceGetInfo function.

Version 1.0.1.0

- UPD: Improved eye detection.

Version 1.0.0.1

- FIX: Memory leaks in reset function.

Version 1.0.0.0

- Initial release.

C.1.2. FccMatcher Library**Version 1.1.0.0**

- UPD: Updated for FccExtractor 1.1 algorithm.

Version 1.0.0.1

- ADD: FccmGetInfo function.

Version 1.0.0.0

- Initial release.

C.1.3. NCore Library**Version 2.4.4.0**

- UPD: Error framework made cross-platform.

Version 2.4.3.0

- ADD: Aligned memory management functions.

Version 2.4.2.1

- FIX: Minor fixes.

Version 2.4.2.0

- ADD: NPointF, NPointD, NSizeF, NSizeD, NRectF and NRectD types.
- ADD: NRange type.
- UPD: Internal optimizations.

Version 2.4.1.1

- FIX: Minor fixes.

Version 2.4.1.0

- ADD: NLibraryInfo module.
- ADD: NCoreGetInfo function.

Version 2.4.0.0

- ADD: Integration with Win32 and COM errors on Windows.
- ADD: NStream module.

Version 2.3.1.0

- ADD: NProcessorInfo module for CPU identification on Windows.

Version 2.3.0.1

- FIX: Memory leak in parameters framework.

Version 2.3.0.0

- ADD: HNStream type.

Version 2.2.2.0

- CHN: NMemory interface.

Version 2.2.1.0

- ADD: More robust error handling on Windows.

Version 2.2.0.0

- ADD: Unicode support.

Version 2.1.0.2

- FIX: Functions' calling convention on Windows.

Version 2.1.0.1

- UPD: Minor updates.

Version 2.1.0.0

- REM: Registration error codes.
- UPD: Updated to use Microsoft Visual C++ Runtime Library 8.0.

Version 2.0.1.1

- CHN: Minor changes.

Version 2.0.1.0

- ADD: [NParameters](#) module instead of NMetaTypes module for internal infrastructure support.
- CHN: Infrastructure optimization for 64-bit support.

Version 2.0.0.0

- ADD: A lot of stuff for internal infrastructure support.
- CHN: Some changes in internal infrastructure support.

Version 1.0.0.2

- ADD: [NIndexPair](#) structure.

Version 1.0.0.1

- FIX: Minor fixes in headers.

Version 1.0.0.0

Initial release.

C.1.4. NImages Library**Version 2.4.0.2**

- FIX: Crash during CMYK JPEG files reading.
- FIX: Memory leak during save in JPEG 2000 format.

Version 2.4.0.1

- UPD: Internal updates.

Version 2.4.0.0

- ADD: PNG format support.
- CHN: RGB to grayscale conversion now uses correct formula.

Version 2.3.0.0

- ADD: Image rotation, flipping and cropping functionality.

Version 2.2.0.2

- ADD: NImagesGetInfo function.

Version 2.2.0.1

- FIX: Some TIFF files reading.

Version 2.2.0.0

- ADD: I/O with HNStream.
- FIX: Some BMP RLE-compressed files reading.

Version 2.1.0.2

- FIX: Saving in JPEG format for some images.

Version 2.1.0.1

- UPD: Minor updates.

Version 2.1.0.0

- ADD: JPEG format support.

Version 2.0.1.2

- FIX: Minor fixes.

Version 2.0.1.1

- FIX: Reading of some BMP files.

Version 2.0.1.0

- ADD: Unicode support.

Version 2.0.0.5

- FIX: Fixed some TIFF files reading.

Version 2.0.0.4

- FIX: Fixed some bad-formed BMP files reading.

Version 2.0.0.3

Initial release.

C.1.5. NTemplate Library

Version 1.4.0.3

- FIX: NERRecordClone does not copy Position.

Version 1.4.0.2

- UPD: Internal updates.

Version 1.4.0.1

- FIX: NTemplateCheck without one or more modality.

Version 1.4.0.0

- ADD: NERRecord and NTemplate modules and irises support in NTemplate module in iris-related products.

Version 1.3.0.0

- UPD: Merged with NFTemplate, NLTemplate, NFRecord and NLRecord libraries.

Version 1.2.0.1

- ADD: NTemplateGetInfo function.

Version 1.2.0.0

- ADD: Faces support.
- UPD: Updated to use Microsoft Visual C++ Runtime Library 8.0.

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.0.3

- UPD: Minor updates.

Version 1.0.0.2

- FIX: Minor fixes.

Version 1.0.0.1

- FIX: Signature of packed NTemplate.
- FIX: NTemplateUnpack function returned error in some cases.
- FIX: NTemplateUnpack logic.

Version 1.0.0.0

Initial release.

C.2. Samples**C.2.1. Windows****C.2.1.1. FaceCell C++ Sample****Version 1.1.0.0**

- CHN: Reference to NLRecord library replaced with reference to NTemplate library.

Version 1.0.1.0

- ADD: Added enrollment from sequence of images.
- ADD: Face quality threshold in options dialog.

Version 1.0.0.2

- FIX: Removed dependencies on Gdi+ when loading images.
- FIX: Minor fixes.

Version 1.0.0.0

- Initial release.