NImages Pro 2.0 Add-On for VeriFinger SDK

NImages Pro 2.0 Add-On for VeriFinger SDK

Published July 12, 2007. Version 2.0.3.0

Table of Contents

Chapter 1. About

1.1. Introduction

NImages Pro Add-On allows to integrate WSQ (Wavelet Scalar Quantization) and NIST IHead image format support into applications based on VeriFinger SDK.

NImages Pro Add-On contains WSQ compression and decompression module, which can be used to minimize storage size of fingerprint images and for data interchange between systems based on MegaMatcher, VeriFinger, FingerCell technologies and other biometric systems.

WSQ fingerprint image compression allows compressing image up to 10-15 times. WSQ compression process is "lossy", meaning that the reconstructed image is not equal to the original (some information is lost). However, the WSQ algorithm was specially designed to minimize the loss of fingerprint information therefore the reconstructed image is as close as possible to the original.

1.2. Platforms Supported

NImages Pro Add-On supports platforms based on x86 processor architecture. Libraries for Windows, Linux and Mac OS X operating systems are provided.

1.3. System Requirements

- PC or Mac with Pentium-compatible 500MHz processor or better.
- Microsoft Windows 9x/ME/NT/2000/XP/2003 or Linux (based on glibc 2.2.5 or compatible) or Mac OS X (10.3.9 or newer).

1.4. Licensing

NImages Pro Add-On license should be obtained only once.

It is allowed to use NImages Pro Add-On components on the PC under which VeriFinger/FingerCell components are used.

Chapter 2. Overview

2.1. Image Support

Image support in the NImages Pro Add-On can be divided into the following four 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.
- Image File. Should be used if multiple images are stored in one file and more than one image should be loaded from the file.
- Low-Level Image Input-Output. Should be used to have more control on how images are loaded and saved in particular format.

2.1.1. 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 NImages Pro Add-On is defined by HNImage handle in NImage Pro module (`NImage` class in .NET). 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 (Width, Height, Stride, PixelFormat and Pixels properties in .NET) for more information.

An image can have horizontal and vertical resolution attributes assigned to it if they are applicable . See `NImageGetHorzResolution` and `NImageGetVertResolution` functions (HorzResolution and VertResolution properties in .NET) for more information.

An image can be created either as empty or from existing memory block. See `NImageCreate`, `NImageCreateFromData` and `NImageCreateWrapper` functions (`Create`, `FromData` and `GetWrapper` methods in .NET) for more information.

For each value of NPixelFormat (NPixelFormat in .NET) exposed via interface a module (subclass of NImage in .NET) is provided for managing according type of image (getting and setting individual pixels, etc.). See NGrayscaleImage, NMonochromeImage and NRgbImage modules (NGrayscaleImage, NMonochromeImage and NRgbImage classes in .NET) for more information.

An image can be converted to different pixel format using `NImageCreateFromImage` or `NImageCreateFromImageEx` function (`FromImage` method in .NET).

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

- On Windows `BmpSaveImageToHBitmap` function (`ToHBitmap` method in .NET) can be used to receive a standard Win32 HBITMAP for the image. The reverse process is also possible using `BmpLoadImageFromHBitmap` function (`FromHBitmap` method in .NET).
- In .NET `ToBitmap` method can be used to receive a standard .NET Bitmap. The reverse process is also possible using `FromBitmap` method.
- On Linux there is no easy method implemented. However, a memory block containing pixels of image could be accessed via `NImageGetPixelFormat` function (PixelFormat method in .NET). 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 (`Save` method in .NET).

An image stored in file in any supported image format can be loaded using `NImageCreateFromFile` function (`FromFile` method in .NET).

Files containing more than one image are also supported. See Image File and Image Format sections for more information.

2.1.2. 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 NImages Pro Add-On is defined by HNImageFormat handle in NImageFormat Pro module (`NImageFormat` class in .NET).

There is a number of image formats supported in the NImages Pro Add-On. 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	In .NET only	In .NET only
NIST IHead	Yes	Yes
JPEG	Yes	Yes
PNG	In .NET only	In .NET only
TIFF	Yes	In .NET only

Image Format	Can read	Can write
WSQ	Yes	Yes

These image formats are accessible using `NImageFormatGetBmp`, `NImageFormatGetIHead`, `NImageFormatGetTiff` and `NImageFormatGetWsq` functions (*`Bmp`*, *`Gif`*, *`IHead`*, *`Jpeg`*, *`Png`* , *`Tiff`* and *`Wsq`* read-only fields in .NET).

To find out which images formats are supported in the NImages Pro Add-On in version-independent way `NImageFormatGetFormatCount` and `NImageFormatGetFormat` functions should be used (Formats property in .NET).

Name, file name pattern (file filter) and default file extension of the image format can be retrieved using `NImageFormatGetName`, `NImageFormatGetFileFilter` and `NImageFormatGetDefaultFileExtension` functions (Name, FileFilter and DefaultFileExtension properties in .NET).

To find out which image format should be used to read or write a particular file `NImageFormatSelect` function (`Select` method in .NET) should be used.

An image can be loaded and saved from/to file or memory buffer using `NImageFormatLoadImageFromFile`, `NImageFormatLoadImageFromMemory`, `NImageFormatSaveImageToFile` and `NImageFormatSaveImageToMemory` functions (`LoadImage` and `SaveImage` methods in .NET). Note that not all image formats support both reading and writing. Use `NImageFormatCanRead` and/or `NImageFormatCanWrite` function(s) (CanRead and/or CanWrite property(ies) in .NET) to check if the particular image format does.

If image file contains more than one image then image file can be opened using `NImageFormatOpenFile` or `NImageFormatOpenFileFromMemory` function (`OpenFile` method in .NET). Image file further can be used to read all images from the file.

If it is needed to store multiple images in one file `NImageFormatSaveImagesToFile` or `NImageFormatSaveImagesToMemory` function (`SaveImages` method in .NET) should be used. Note that not all image formats support writing of multiple images. Use `NImageFormatCanWriteMultiple` function (CanWriteMultiple property in .NET) to check if the particular image format supports.

2.1.3. Image File

Image file in the NImages Pro Add-On is defined by HNImageFile handle in NImageFile module (`NImageFile` class in .NET) is an encapsulation of an opened read-only file containing one or more images.

An image file is opened using `NImageFileCreate` function (`FromFile` method in .NET). Also an image file can be opened using image format.

Images are read from image file subsequently calling `NImageFileReadImage` function

`ReadImage` method in .NET) until HNImage (NImage in .NET) it reads is NULL (`null` in .NET).

2.1.4. Low-Level Image Input-Output

Low-level image I/O in the NImages Pro Add-On is implemented in Bmp, IHead, Tiff and Wsq modules (`Bmp`, `IHead`, `Tiff` and `Wsq` classes in .NET).

These modules (classes in .NET) provides functions (static methods in .NET) for loading and saving images in according format (BMP, NIST IHead, TIFF and WSQ).

Those functions (static methods in .NET) can take parameters that precisely control loading and saving of the image in particular formats. For example, bit rate is specified when saving in WSQ format.

Chapter 3. Using

To use NImages Pro Add-On with VeriFinger SDK copy files from `bin`, `include` and `lib` directories to according directories in VeriFinger 5.0 SDK. Refer to this documentation and to VeriFinger SDK samples on how to load, save and use images.

Chapter 4. Reference (C/C++)

This chapter contains reference of all libraries included in NImages Pro Add-On for developers using C/C++ language.

Libraries

NCore	Provides infrastructure for Neurotechnologija components.
NImages Pro	Provides functionality for loading, saving and converting images in various formats.

4.1. NCore Library

Provides infrastructure for Neurotechnologija components.

Windows

Import library: `NCore.dll.lib`.

DLL: `NCore.dll`.

Requirements:

- Microsoft Visual C++ 2005 Libraries runtime.
- Microsoft Layer for Unicode on Windows 98/ME (`unicows.dll`).

Linux

Shared object: `libNCore.so`.

Mac OS X

Shared object: `libNCore.dylib`.

Modules

NCore	Provides infrastructure/basic functionality for Neurotechnologija components.
NErrors	Defines error codes used in Neurotechnologija components.
NMemory	Provides memory management for Neurotechnologija components.

NParameters	Provides functionality for working with parameters for Neurotechnologija components.
NTypes	Defines types and macros used in Neurotechnologija components.

4.1.1. NCore Module

Provides infrastructure/basic functionality for Neurotechnologija components.

Header file: `NCore.h`.

See Also

NCore Library

4.1.2. NErrors Module

Defines error codes used in Neurotechnologija components.

Header file: `NErrors.h`.

Macros

-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.
-2	N_E_CORE	Standard error has occurred (for internal use).
-15	N_E_END_OF_STREAM	Attempted to read file or buffer after its end.
-1	N_E_FAILED	Unspecified error has occurred.
-13	N_E_FORMAT	Format of argument value is invalid.
-9	N_E_INDEX_OUT_OF_RANGE	Index is out of range (for internal use).
-7	N_E_INVALID_OPERATION	Attempted to perform invalid operation.
-14	N_E_IO	Input/output error has occurred.

-5	N_E_NOT_IMPLEMENTED	Functionality is not implemented.
-200	N_E_NOT_REGISTERED	Module is not registered.
-6	N_E_NOT_SUPPORTED	Functionality is not supported.
-3	N_E_NULL_REFERENCE	Null reference has occurred (for internal use).
-4	N_E_OUT_OF_MEMORY	There were not enough memory.
-8	N_E_OVERFLOW	Arithmetic overflow has occurred.
-100	N_E_PARAMETER	Parameter ID is invalid.
-101	N_E_PARAMETER_READ_ONLY	Attempted to set read only parameter.
0	N_OK	No error.
	NFailed	Determines whether function result indicates error.
	NSucceeded	Determines whether function result indicates success.

See Also

NCore Library

4.1.3. NMemory Module

Provides memory management for Neurotechnologija components.

Header file: `NMemory.h`.

Functions

`NAlloc`	Allocates memory block.
`NCAlloc`	Allocates memory block with all bytes set to zero.
`NCompare`	Compares bytes in two memory blocks.
`NCopy`	Copies data between memory blocks.
`NFill`	Sets bytes of memory block to specified value.
`NFree`	Deallocates memory block.

`NMove`	Move data from one memory block to another.
`NReAlloc`	Reallocates memory block.

Macros

NClear	Clears memory block.

See Also

NCore Library

4.1.4. NParameters Module

Provides functionality for working with parameters for Neurotechnologija components.

Header file: `NParameters.h`.

Macros

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

N_TYPE_UINT	Specifies that parameter type is NUInt.
N_TYPE_ULONG	Specifies that parameter type is NULong.
N_TYPE_USHORT	Specifies that parameter type is NUShort.

See Also

NCore Library

4.1.4.1. NParameterMakeId Macro

Makes parameter id.

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

Parameters

code	One of N_PC_XXX.
index	Reserved, must be zero.
id	One of the parameter ids provided by a Neurotechnologija module.

See Also

NParameters Module

4.1.5. NTypes Module

Defines types and macros used in Neurotechnologija 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

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

	port.
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

4.1.5.1. 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

4.1.5.2. 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

4.1.5.3. 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

4.1.5.3.1. NIndexPair.Index1 Field

First index of this NIndexPair.

```
NInt Index1;
```

See Also

NIndexPair

4.1.5.3.2. NIndexPair.Index2 Field

Second index of this NIndexPair.

```
NInt Index2;
```

See Also

NIndexPair

4.1.5.4. NRational Structure

Represents a signed rational number.

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

Fields

Denominator	Denominator of this NRational.
Numerator	Numerator of this NRational.

See Also

NTypes Module

4.1.5.4.1. NRational.Denominator Field

Denominator of this NRational.

```
NInt Denominator;
```

See Also

NRational

4.1.5.4.2. NRational.Numerator Field

Numerator of this NRational.

```
NInt Numerator;
```

See Also

NRational

4.1.5.5. NURational Structure

Represents an unsigned rational number.

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

Fields

Denominator	Denominator of this NURational.
Numerator	Numerator of this NURational.

See Also

NTypes Module

4.1.5.5.1. NURational.Denominator Field

Denominator of this NURational.

```
NUInt Denominator;
```

See Also

NURational

4.1.5.5.2. NURational.Numerator Field

Numerator of this NURational.

```
NUInt Numerator;
```

See Also

NURational

4.1.6. NGeometry Module

Provides definitions of geometrical structures types.

Header file: `NGeometry.h` (includes `NTypes.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

4.1.6.1. 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

4.1.6.1.1. NPoint.X Field

Point coordinate on x axis.

```
NInt X;
```

See Also

NPoint

4.1.6.1.2. NPoint.Y Field

Point coordinate on y axis.

```
NInt Y;
```

See Also

NPoint

4.1.6.2. NSize structure

Structure defining rectangle size.

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

Fields

Width	Width.
Height	Height.

See Also

NGeometry

4.1.6.2.1. NSize.Width Field

Width.

```
NInt Width;
```

See Also

NSize

4.1.6.2.2. NSize.Height Field

Height.

```
NInt Height;
```

See Also

NSize

4.1.6.3. NRect structure

Structure defining a rectangle figure in 2D space.

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

Fields

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

See Also

NGeometry

4.1.6.3.1. NRect.X Field

Upper left rectangle corner coordinate on x axis.

```
NInt X;
```

See Also

NRect

4.1.6.3.2. NRect.Y Field

Upper left rectangle corner coordinate on y axis.

```
NInt Y;
```

See Also

NRect

4.1.6.3.3. NRect.Width Field

Rectangle width.

```
NInt Width;
```

See Also

NRect

4.1.6.3.4. NRect.Height Field

Rectangle height.

```
NInt Height;
```

See Also

NRect

4.2. NImages Pro Library

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

Windows

Import library: `NImages.dll.lib`.

DLL: `NImages.dll`.

Requirements:

- `NCore.dll`.

Linux

Shared object: `libNImages.so`.

Requirements:

- `libNCore.so`.

Mac OS X

Shared object: `libNImages.dylib`.

Requirements:

- `libNCore.dylib`.

Modules

Bmp	Provides functionality for loading and saving images in BMP format.
IHead	Provides functionality for loading and saving images in NIST IHead format.
Jpeg	Provides functionality for loading and saving images in JPEG format.
NGrayscaleImage	Provides functionality for managing 8-bit grayscale images.
NImageFile	Provides functionality for reading image files in format-neutral way.
NImageFormat Pro	Provides functionality for loading and saving images in format-neutral way.
NImage Pro	Provides functionality for managing images.

NImages Pro	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.
Tiff	Provides functionality for loading images in TIFF format.
Wsq	Provides functionality for loading and saving images in WSQ format.

4.2.1. 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 Pro Library

4.2.1.1. BmpLoadImageFromFile Function

Loads image from BMP file.


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

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[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	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid.

Remarks

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

See Also

Bmp Module | HNImage | BmpLoadImageFromMemory | BmpLoadImageFromHBitmap | BmpSaveImageToFile

4.2.1.2. BmpLoadImageFromHBitmap Function

> **Note**
> This function is available only on Windows.

Loads image from Windows HBITMAP.

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

Parameters

handle	[in] Handle that specifies Windows HBITMAP.
pHImage	[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	*handle* or *pHImage* 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`

4.2.1.3. BmpLoadImageFromMemory Function

Loads image from memory buffer containing BMP file.

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

Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[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	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* 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

4.2.1.4. BmpSaveImageToFile Function

Saves image to file in BMP format.

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

Parameters

hImage	[in] Handle to image.
szFileName	[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	*hImage* or *szFileName* 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`

4.2.1.5. BmpSaveImageToHBitmap Function

Note
This function is available only on Windows.

Saves image to Windows HBITMAP.

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

Parameters

hImage	[in] Handle to image.
pHandle	[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	*hImage* or *pHandle* is NULL.

Remarks

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

See Also

Bmp Module | HNImage | `BmpSaveImageToFile` | `BmpSaveImageToMemory` `BmpLoadImageFromHBitmap`

4.2.1.6. BmpSaveImageToMemory Function

Saves image to memory buffer in BMP format.

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

Parameters

hImage	[in] Handle to image.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[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	*hImage*, *pBuffer* or *pBufferLength* 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`

4.2.2. IHead Module

Provides functionality for loading and saving images in NIST IHead format.

Header file: `IHead.h`.

Functions

`IHeadLoadImageFromFile`	Loads image from NIST IHead file.
`IHeadLoadImageFromMemory`	Loads image from memory buffer containing NIST IHead file.
`IHeadSaveImageToFile`	Saves image to the file in NIST IHead format.
`IHeadSaveImageToMemory`	Saves image to the memory buffer in NIST IHead format.

See Also

NImages Pro Library

4.2.2.1. IHeadLoadImageFromFile Function

Loads image from NIST IHead file.

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

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[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	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid.

Remarks

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

See Also

IHead Module | HNImage | IHeadLoadImageFromMemory

4.2.2.2. IHeadLoadImageFromMemory Function

Loads image from memory buffer containing NIST IHead file.

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

Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[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	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid.

Remarks

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

See Also

IHead Module | HNImage | IHeadLoadImageFromFile

4.2.2.3. IHeadSaveImageToFile Function

Saves image to the file in NIST IHead format.

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

Parameters

hImage	[in] Handle to image.
szFileName	[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	*hImage* or *szFileName* is NULL.

Remarks

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

See Also

IHead Module | HNImage | `IHeadSaveImageToMemory` | `IHeadLoadImageFrom-File`

4.2.2.4. IHeadSaveImageToMemory Function

Saves image to the memory buffer in NIST IHead format.

```
NResult N_API IHeadSaveImageToMemory(
        HNImage hImage,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImage	[in] Handle to image.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[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	*hImage*, *pBuffer* or *pBufferLength* 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

IHead Module | HNImage | `IHeadSaveImageToFile` | `IHeadLoadImageFrom-Memory`

4.2.3. 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

JPEG_DEFAULT_QUALITY	Specifies default JPEG quality.

See Also

NImages Pro Library

4.2.3.1. JpegLoadImageFromFile Function

Loads image from JPEG file.

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

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[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	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid.

See Also

Jpeg Module | HNImage | JpegLoadImageFromMemory | JpegSaveImageToFile

4.2.3.2. JpegLoadImageFromMemory Function

Loads image from memory buffer containing JPEG file.

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

Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[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	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid.

See Also

Jpeg Module | HNImage | JpegLoadImageFromFile | JpegSaveImageToMemory

4.2.3.3. JpegSaveImageToFile Function

Saves image to file in JPEG format with specified bitrate.

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

Parameters

hImage	[in] Handle to image.
quality	[in] Specifies quality of JPEG image.
szFileName	[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	*hImage* or *szFileName* is NULL.

See Also

Jpeg Module | HNImage | JPEG_DEFAULT_QUALITY | `JpegSaveImageToMemory` | `JpegLoadImageFromFile`

4.2.3.4. 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

hImage	[in] Handle to image.
quality	[in] Specifies quality of JPEG image.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[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	*hImage*, *pBuffer* or *pBufferLength* 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`

4.2.4. NGrayscaleImage Module

Provides functionality for managing 8-bit grayscale images.

Header file: `NGrayscaleImage.h`.

Functions

`NGrayscaleImageGetPixel`	Retrieves value of pixel at the specified co-ordinates 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 Pro Library | NImage Pro Module

4.2.4.1. 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

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
pValue	[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	*hImage* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*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 `npf-Grayscale`.

See Also

NGrayscaleImage Module | HNImage | `NGrayscaleImageSetPixel`

4.2.4.2. 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

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
value	[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	*hImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*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 npf-Grayscale.

See Also

NGrayscaleImage Module | HNImage | NGrayscaleImageGetPixel

4.2.5. NImageFile Module

Provides functionality for reading image files in format-neutral way.

Header file: NImageFile.h.

Functions

NImageFileClose	Closes the file associated with the image file.
NImageFileCreate	Opens image file of specified format.
NImageFileFree	Closes the image file. After the image file is closed the specified handle is no longer valid.
NImageFileGetFormat	Retrieves image format of the image file.
NImageFileIsOpened	Retrieves a value indicating whether the file associated with the image file is opened.
NImageFileReadImage	Reads image from the image file.

Types

HNImageFile	Handle to opened read-only image file.

See Also

NImages Pro Library | NImageFormat Pro Module | NImage Pro Module

4.2.5.1. NImageFileClose Function

Closes the file associated with the image file.

```
NResult N_API NImageFileClose(
        HNImageFile hImageFile
);
```

Parameters

hImageFile	[in] Handle to image file.

Return Values

If the function succeeds, the return 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	*hImageFile* is NULL.

Remarks

If the file associated with the image file is already closed does nothing.

See Also

NImageFile Module | HNImageFile | `NImageFileIsOpened` | `NImageFileFree`

4.2.5.2. NImageFileCreate Function

Opens image file of specified format.

```
NResult N_API NImageFileCreate(
        const NChar * szFileName,
```



```
        HNImageFormat hImageFormat,
        HNImageFile * pHImageFile
);
```

Parameters

szFileName	[in] Points to string that specifies file name.
hImageFormat	[in] Handle to the image format of the file. Can be NULL.
pHImageFile	[out] Pointer to HNImageFile that receives handle to opened image file.

Return Values

If the function succeeds, the return 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	*szFileName* or *pHImageFile* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid for specified image format.
N_E_NOT_SUPPORTED	*hImageFormat* is NULL and none of supported image formats is registered with file extension of *szFileName*. - or - *hImageFormat* is NULL and image format registered with file extension of *szFileName* does not support reading. - or - Image format specified by *hImageFormat* does not support reading.

Remarks

If *hImageFormat* is NULL image format is selected by file extension of *szFileName*.

Opened image file must be closed using NImageFileFree function.

This function does not check format of the file for all image formats. However format of the file is always checked in `NImageFileReadImage` function.

See Also

NImageFile Module | HNImageFile | `NImageFileFree` | `NImageFileReadImage`

4.2.5.3. NImageFileFree Function

Closes the image file. After the image file is closed the specified handle is no longer valid.

```
void N_API NImageFileFree(
        HNImageFile hImageFile
);
```

Parameters

hImageFile	[in] Handle to image file.

Remarks

If *hImageFile* is NULL does nothing.

See Also

NImageFile Module | HNImageFile | `NImageFileCreate` | `NImageFileClose`

4.2.5.4. NImageFileGetFormat Function

Retrieves image format of the image file.

```
NResult N_API NImageFileGetFormat(
        HNImageFile hImageFile,
        HNImageFormat * pValue
);
```

Parameters

hImageFile	[in] Handle to image file.
pValue	[out] Pointer to NImageFormat that receives image format of the image file.

Return Values

If the function succeeds, the return 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	*hImageFile* or *pValue* is NULL.

See Also

NImageFile Module | HNImageFile

4.2.5.5. NImageFileIsOpened Function

Retrieves a value indicating whether the file associated with the image file is opened.

```
NResult N_API NImageFileIsOpened(
        HNImageFile hImageFile,
        NBool * pValue
);
```

Parameters

hImageFile	[in] Handle to image file.
pValue	[out] Pointer to NBool that receives value indicating whether the file associated with the image file is opened.

Return Values

If the function succeeds, the return 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	*hImageFile* or *pValue* is NULL.

See Also

NImageFile Module | HNImageFile | `NImageFileClose`

4.2.5.6. NImageFileReadImage Function

Reads image from the image file.

```
NResult N_API NImageFileReadImage(
        HNImageFile hImageFile,
        HNImage * pHImage
);
```

Parameters

hImageFile	[in] Handle to image file.
pHImage	[out] Pointer to HNImage that receives handle to image read from the image file.

Return Values

If the function succeeds, the return 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	*hImageFile* or *pHImage* is NULL.
N_E_FORMAT	Format of file associated with the image file is invalid for the image format of the image file.
N_E_INVALID_OPERATION	File associated with the image file is closed.

Remarks

If all images are already read from the image file then handle to image returned via *pHImage* is NULL.

See Also

NImageFile Module | HNImageFile | HNImage | `NImageFileIsOpened`

4.2.6. NImageFormat Pro Module

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

Header file: `NImageFormatPro.h` (includes `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.
NImageFormatCanWriteMultiple	Retrieves a value indicating whether the image format supports writing of multiple images.
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.
NImageFormatGetIHead	Retrieves NIST IHead image format.
NImageFormatGetName	Retrieves name of the image format.
NImageFormatGetTiff	Retrieves TIFF image format.
NImageFormatGetWsq	Retrieves WSQ image format.
NImageFormatLoadImageFromFile	Loads image from file of specified image format.
NImageFormatLoadImageFromMemory	Loads image from the memory buffer containing file of specified image format.
NImageFormatOpenFile	Opens image file of specified format.
NImageFormatOpenFileFromMemory	Opens image file from the memory buffer containing file of specified format.
NImageFormatSaveImageToFile	Saves image to the file in specified format.
NImageFormatSaveImageToMemory	Saves image to the memory buffer in specified format.
NImageFormatSaveImagesToFile	Saves array of images to the file in specified format.
NImageFormatSaveImagesToMemory	Saves array of images 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 Pro Library | NImage Pro Module | NImageFile Module

4.2.6.1. NImageFormatCanRead Function

Retrieves a value indicating whether the image format supports reading.

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

Parameters

hImageFormat	[in] Handle to image format.
pValue	[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	*hImageFormat* or *pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanWrite

4.2.6.2. NImageFormatCanWrite Function

Retrieves a value indicating whether the image format supports writing.

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

Parameters

hImageFormat	[in] Handle to image format.
pValue	[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	*hImageFormat* or *pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | `NImageFormatCanRead` | `NImageFormatCanWriteMultiple`

4.2.6.3. NImageFormatCanWriteMultiple Function

Retrieves a value indicating whether the image format supports writing of multiple images.

```
NResult N_API NImageFormatCanWriteMultiple(
        HNImageFormat hImageFormat,
        NBool * pValue
);
```

Parameters

hImageFormat	[in] Handle to image format.
pValue	[out] Pointer to NBool that receives value

	indicating whether the image format supports writing of multiple images.

Return Values

If the function succeeds, the return 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	*hImageFormat* or *pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | NImageFormatCanWrite

4.2.6.4. NImageFormatGetBmp Function

Retrieves BMP image format.

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

Parameters

pValue	[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	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatGetIHead | NImage-FormatGetTiff | NImageFormatGetWsq

4.2.6.5. NImageFormatGetDefaultFileExtension Function

Retrieves default file extension of the image format.

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

Parameters

hImageFormat	[in] Handle to image format.
pValue	[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	*hImageFormat* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat

4.2.6.6. NImageFormatGetFileFilter Function

Retrieves file filter of the image format.

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


Parameters

hImageFormat	[in] Handle to image format.
pValue	[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	*hImageFormat* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat

4.2.6.7. NImageFormatGetFormat Function

Retrieves supported image format with specified index.

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

Parameters

index	[in] Specifies zero-based supported image format index to retrieve.
pValue	[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	*pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to supported image format count. See NImageFormatGetFormatCount.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatGetFormatCount

4.2.6.8. NImageFormatGetFormatCount Function

Retrieves number of supported image formats.

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

Parameters

pValue	[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	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatGetFormat

4.2.6.9. NImageFormatGetIHead Function

Retrieves NIST IHead image format.


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

Parameters

pValue	[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	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | `NImageFormatGetBmp` | `NImageFormatGetTiff` | `NImageFormatGetWsq`

4.2.6.10. NImageFormatGetName Function

Retrieves name of the image format.

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

Parameters

hImageFormat	[in] Handle to image format.
pValue	[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	*hImageFormat* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat

4.2.6.11. NImageFormatGetTiff Function

Retrieves TIFF image format.

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

Parameters

pValue	[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	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | `NImageFormatGetBmp` | `NImageFormatGetIHead` | `NImageFormatGetWsq`

4.2.6.12. NImageFormatGetWsq Function

Retrieves WSQ image format.


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

Parameters

pValue	[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	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | `NImageFormatBmp` | `NImageFormatGetIHead` | `NImageFormatGetTiff`

4.2.6.13. NImageFormatLoadImageFromFile Function

Loads image from file of specified image format.

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

Parameters

hImageFormat	[in] Handle to image format.
szFileName	[in] Points to string that specifies file name.
pHImage	[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	*hImageFormat*, *szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support reading.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | HNImage | NImageFormatLoadImageFromMemory | NImageFormatSaveImageToFile | NImageFormatOpenFile

4.2.6.14. 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

hImageFormat	[in] Handle to image format.
buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[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	*hImageFormat* or *pHImage* is NULL. - or - *buffer* is NULL and *bufferLength* is not equal to zero.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by *hImage-Format* does not support reading.

See Also

NImageFormat Pro Module | HNImageFormat | `NImageFormatCanRead` | HNImage | `NImageFormatLoadImageFromFile` | `NImageFormatSaveImageToMemory` | `NImageFormatOpenFileFromMemory`

4.2.6.15. NImageFormatOpenFile Function

Opens image file of specified format.

```
NResult N_API NImageFormatOpenFile(
        HNImageFormat hImageFormat,
        const NChar * szFileName,
        HNImageFile * pHImageFile
);
```

Parameters

hImageFormat	[in] Handle to image format.
szFileName	[in] Points to string that specifies filename.
pHImageFile	[in] Pointer to HNImageFile that receives handle to opened image file.

Return Values

If the function succeeds, the return 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	*hImageFormat*, *szFileName* or *pHImageFile* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support reading.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | HNImageFile | NImageFormatOpenFileFromMemory | NImageFormatLoadImageFromFile

4.2.6.16. NImageFormatOpenFileFromMemory Function

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

```
NResult N_API NImageFormatOpenFileFromMemory(
        HNImageFormat hImageFormat,
        const void * buffer,
        NSizeType bufferLength,
        HNImageFile * pHImageFile
);
```

Parameters

hImageFormat	[in] Handle to image format.
buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImageFile	[out] Pointer to HNImageFile that receives handle to opened image file.

Return Values

If the function succeeds, the return 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	*hImageFormat* or *pHImageFile* is NULL. - or - *buffer* is NULL and *bufferLength* is not equal to zero.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by *hImage-Format* does not support reading.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | HNImageFile | NImageFormatOpenFile | NImageFormatLoadImageFromMemory

4.2.6.17. NImageFormatSaveImagesToFile Function

Saves array of images to the file in specified format.

```
NResult N_API NImageFormatSaveImagesToFile(
        HNImageFormat hImageFormat,
        NInt imageCount,
        HNImage * arHImages,
        const NChar * szFileName
);
```

Parameters

hImageFormat	[in] Handle to image format.
imageCount	[in] Specifies image count in the array.
arHImages	[in] Points to array of handles to images.
szFileName	[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	*hImageFormat*, *arHImages* or *szFileName* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*imageCount* is less than or equal to zero.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support writing. - or - *imageCount* is greater than one and image format specified by *hImageFormat* does not support writing of multiple files.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanWriteMultiple | HNImage | NImageFormatSaveImagesToMemory | NImageFormatSaveImageToFile | NImageFormatLoadImageFromFile | NImageFormatOpenFile

4.2.6.18. NImageFormatSaveImagesToMemory Function

Saves array of images to the memory buffer in specified format.

```
NResult N_API NImageFormatSaveImagesToMemory(
        HNImageFormat hImageFormat,
        NInt imageCount,
        HNImage * arHImages,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImageFormat	[in] Handle to image format.
imageCount	[in] Specifies image count in the array.
arHImages	[in] Points to array of handles to images.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[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	*hImageFormat*, *arHImages*, *pBuffer* or *pBufferLength* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*imageCount* is less than or equal to zero.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support writing. - or - *imageCount* is greater than one and image format specified by *hImageFormat* does not support writing of multiple files.

Remarks

Memory buffer allocated by the function must be deallocated using NFree function when it is no longer needed.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanWriteMultiple | HNImage | NImageFormatSaveImagesToFile | NImageFormatSaveImageToMemory | NImageFormatLoadImageFromMemory | NImageFormatOpenFileFromMemory

4.2.6.19. NImageFormatSaveImageToFile Function

Saves image to the file in specified format.

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

Parameters

hImageFormat	[in] Handle to image format.

hImage	[in] Handle to image.
szFileName	[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	*hImageFormat*, *hImage* or *szFileName* is NULL.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support writing.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanWrite | HNImage | NImageFormatSaveImageToMemory | NImageFormatLoadImageFromFile

4.2.6.20. NImageFormatSaveImageToMemory Function

Saves image to the memory buffer in specified format.

```
NResult N_API NImageFormatSaveImageToMemory(
        HNImageFormat hImageFormat,
        HNImage hImage,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImageFormat	[in] Handle to image format.
hImage	[in] Handle to image.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[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	*hImageFormat*, *hImage*, *pBuffer* or *pBufferLength* is NULL.
N_E_NOT_SUPPORTED	Image format specified by *hImage-Format* 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 Pro Module | HNImageFormat | NImageFormatCanWrite | HNImage | NImageFormatSaveImageToFile | NImageFormatLoadImageFromMemory

4.2.6.21. 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,
        HNImageFormat * pHImageFormat
);
```

Parameters

szFileName	[in] Points to string that file name.
fileAccess	[in] Specifies that image format should support reading, writing or both.
pHImageFormat	[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	*fileAccess* value is invalid.
N_E_ARGUMENT_NULL	*szFileName* or *pHImageFormat* 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 Pro Module | HNImageFormat | NFileAccess | NImageFormatGetFormatCount | NImageFormatGetFormat

4.2.7. NImage Pro Module

Provides functionality for managing images.

Header file: NImagePro.h (includes 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 Pro Library | NMonochromeImage Module | NGrayscaleImage Module | NRgbImage Module | NImageFormat Pro Module | NImageFile Module

4.2.7.1. NImageClone Function

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

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


Parameters

hImage	[in] Handle to the image.
pHClonedImage	[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	*hImage* or *pHClonedImage* is NULL.

Remarks

Created image must be deleted using `NImageFree` function.

See Also

NImage Pro Module | HNImage | `NImageFree` | `NImageCreate`

4.2.7.2. 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

pixelFormat	[in] Specifies pixel format of the image.
width	[in] Specifies width of the image.
height	[in] Specifies height of the image.

stride	[in] Specifies stride of the image. Can be zero.
horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
pHImage	[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	*pixelFormat* has invalid value. - or - *stride* is not zero and is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	*pHImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* 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 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 Pro Module | HNImage | `NImageFree` | `NImageCreateWrapper` | `NImageCreateFromData` | `NImageCreateFromImage` | `NImageCreateFromFile` | `NImageClone` | `NImageGetStride`

4.2.7.3. 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

pixelFormat	[in] Specifies pixel format of the image.
width	[in] Specifies width of the image.
height	[in] Specifies height of the image.
stride	[in] Specifies stride of the image. Can be zero.
horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
srcStride	[in] Specifies stride of pixels to be copied to the image.
srcPixels	[in] Points to memory block containing pixels that to be copied to the image.
pHImage	[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	*pixelFormat* has invalid value. - or - *stride* is not zero and is less than minimal value for specified pixel format and width. - or - *srcStride* is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	*srcPixels* or *pHImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is less than zero.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride 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 Pro Module | HNImage | NImageFree | NImageCreate | NImageCreateWrapper | NImageGetStride | NImageGetPixels

4.2.7.4. NImageCreateFromFile Function

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

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

Parameters

szFileName	[in] Points to string that specifies file name.
hImageFormat	[in] Handle to the image format of the file. Can be NULL.
pHImage	[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	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid for specified image format.
N_E_NOT_SUPPORTED	*hImageFormat* is NULL and none of supported image formats is registered with file extension of *szFileName*. - or - *hImageFormat* is NULL and image format registered with file extension of *szFileName* does not support reading. - or - Image format specified by *hImageFormat* does not support reading.

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 Pro Module | HNImage | NImageFree | NImageCreate | NImageFormat-CanRead

4.2.7.5. 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

pixelFormat	[in] Specifies pixel format of the image.
stride	[in] Specifies stride of the image. Can be zero.
hSrcImage	[in] Handle to image used as source for the image.
pHImage	[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	*pixelFormat* has invalid value. - or - *stride* is not zero and is less than minimal value for specified pixel format and source image width.

Error Code	Condition
N_E_ARGUMENT_NULL	*hSrcImage* or *pHImage* is NULL.

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.

See Also

NImage Pro Module | HNImage | NImageFree | NImageCreate | NImageCreateFromImageEx | NImageClone | NImageGetStride

4.2.7.6. 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

pixelFormat	[in] Specifies pixel format of the image.
stride	[in] Specifies stride of the image. Can be zero.
horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
hSrcImage	[in] Handle to image used as source for the image.
pHImage	[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	*pixelFormat* has invalid value. - or - *stride* is not zero and is less than minimal value for specified pixel format and source image width.
N_E_ARGUMENT_NULL	*hSrcImage* or *pHImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*horzResolution* or *vertResolution* 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 Pro Module | HNImage | NImageFree | NImageCreate | NImageCreateFromImage | NImageClone | NImageGetStride

4.2.7.7. 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

pixelFormat	[in] Specifies pixel format of the image.
width	[in] Specifies width of the image.
height	[in] Specifies height of the image.
stride	[in] Specifies stride of the image.
horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
pixels	[in] Points to memory block containing pixels for the image.
ownsPixels	[in] Specifies whether pixels will be automatically deleted with the image (if set to NTrue).
pHImage	[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	*pixelFormat* has invalid value. - or - *stride* is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	*pixels* or *pHImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero.

Error Code	Condition
	- or - *horzResolution* or *vertResolution* 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 Pro Module | HNImage | NImageFree | NImageCreate | NImageCreateFromData | NImageGetStride | NImageGetPixels

4.2.7.8. NImageFree Function

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

```
void N_API NImageFree(
        HNImage hImage
);
```

Parameters

hImage	[in] Handle to the image.

Remarks

If *hImage* is NULL does nothing.

See Also

NImage Pro Module | HNImage | NImageCreate

4.2.7.9. NImageGetHeight Function

Retrieves height of the image.

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

Parameters

hImage	[in] Handle to the image.
pValue	[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	*hImage* or *pValue* is NULL.

See Also

NImage Pro Module | HNImage | `NImageGetWidth`

4.2.7.10. NImageGetHorzResolution Function

Retrieves horizontal resolution of the image.

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

Parameters

hImage	[in] Handle to the image.
pValue	[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	*hImage* or *pValue* is NULL.

Remarks

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

See Also

NImage Pro Module | HNImage | NImageGetVertResolution

4.2.7.11. NImageGetPixelFormat Function

Retrieves pixel format of the image.

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

Parameters

hImage	[in] Handle to the image.
pValue	[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	*hImage* or *pValue* is NULL.

See Also

NImage Pro Module | HNImage | NPixelFormat

4.2.7.12. NImageGetPixels Function

Retrieves pointer to memory block containing pixels of the image.

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

Parameters

hImage	[in] Handle to the image.
pValue	[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	*hImage* or *pValue* 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 Pro Module | HNImage | `NImageGetPixelFormat` | `NImageGetWidth` | `NImageGetHeight` | `NImageGetStride` | `NImageGetSize`

4.2.7.13. NImageGetSize Function

Retrieves size of memory block containing pixels of the image.

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

Parameters

hImage	[in] Handle to the image.
pValue	[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	*hImage* or *pValue* 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 Pro Module | HNImage | `NImageGetHeight` | `NImageGetStride`

4.2.7.14. NImageGetStride Function

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

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


Parameters

hImage	[in] Handle to the image.
pValue	[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	*hImage* or *pValue* is NULL.

Remarks

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

See Also

NImage Pro Module | HNImage | NPixelFormatGetRowSize | `NImageGetPixelFormat` | `NImageGetWidth` | `NImageGetSize`

4.2.7.15. NImageGetVertResolution Function

Retrieves vertical resolution of the image.

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

Parameters

hImage	[in] Handle to the image.
pValue	[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	*hImage* or *pValue* is NULL.

Remarks

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

See Also

NImage Pro Module | HNImage | `NImageGetHorzResolution`

4.2.7.16. NImageGetWidth Function

Retrieves width of the image.

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

Parameters

hImage	[in] Handle to the image.
pValue	[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	*hImage* or *pValue* is NULL.

See Also

NImage Pro Module | HNImage | `NImageGetHeight` | `NImageGetStride`

4.2.7.17. NImageSaveToFile Function

Saves the image to the file of specified format.

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

Parameters

hImage	[in] Handle to NImage object.
szFileName	[in] Points to string that specifies file name.
hImageFormat	[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	*hImage* or *szFileName* is NULL.
N_E_NOT_SUPPORTED	*hImageFormat* is NULL and none of supported image formats is registered with file extension of *szFileName*. - or - *hImageFormat* is NULL and image format registered with file extension of *szFileName* does not support writing. - or - Image format specified by *hImageFormat* does not support writing.

Remarks

If *hImageFormat* is NULL image format is selected by file extension of *szFileName*.

See Also

NImage Pro Module | HNImage | NImageCreateFromFile | NImageFormatCanWrite

4.2.8. NImages Pro Module

Provides library registration and other additional functionality.

Header file: NImagesPro.h (includes NImages.h).

Functions

NImagesGetGrayscaleColorWrapper	Creates color wrapper for grayscale image.
NImagesIsRegistered	Checks if NImages Pro library is registered.

See Also

NImages Pro Library

4.2.8.1. NImagesGetGrayscaleColorWrapper Function

Creates color wrapper for grayscale image.

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

Parameters

hImage	[in] Handle to image.
minColor	[in] Specifies color to be used for black color.
maxColor	[in] Specifies color to be used for white color.
pHDstImage	[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 *hImage* has non-grayscale pixel format (not npfGrayscale or npfMonochrome).
N_E_ARGUMENT_NULL	*hImage* or *pHDstImage* 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 Pro Module | HNImage | NImageFree

4.2.8.2. NImagesIsRegistered Function

Checks if NImages Pro library is registered.

```
NBool N_API NImagesIsRegistered(void);
```

Return Values

NTrue if library is registered, NFalse otherwise.

See Also

NImages Pro Module

4.2.9. 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 Pro Library | NImage Pro Module

4.2.9.1. 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

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
pValue	[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	*hImage* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*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 npf-Monochrome.

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

4.2.9.2. 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

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
value	[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	*hImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*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 `npf-Monochrome`.

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`

4.2.10. NPixelFormat Module

Provides functionality for working with image pixel format.

Header file: `NPixelFormat.h`.

Functions

`NPixelFormatGetBitsPerPixel-Func`	Used internally in NPixelFormatGetBitsPer-Pixel macro.
`NPixelFormatIsValid`	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

NCalcRowSize	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
NCalcRowSizeEx	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel and alignment.
NPixelFormatGetBitsPerPixel	Retrieves number of bits used to store a pixel from NPixelFormat.
NPixelFormatGetRowSize	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.
NPixelFormatGetRowSizeEx	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat and alignment.
NRgbConst	Makes NRgb constant with field values provided.

See Also

NImages Pro Library

4.2.10.1. NPixelFormat Enumeration

Specifies pixel format of each pixel in the image.

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

Members

`npfGrayscale`	Each pixel value is stored in 8 bits representing 256 shades of gray.
`npfMonochrome`	Each pixel value is stored in 1 bit representing either black or white color.
`npfRgb`	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

4.2.10.2. 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

4.2.10.2.1. NRgb.Blue Field

Blue component value of this NRgb.

```
NByte Blue;
```

See Also

NRgb Structure

4.2.10.2.2. NRgb.Green Field

Green component value of this NRgb.

```
NByte Green;
```


See Also

NRgb Structure

4.2.10.2.3. NRgb.Red Field

Red component value of this NRgb.

```
NByte Red;
```

See Also

NRgb Structure

4.2.11. 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 Pro Library | NImage Pro Module

4.2.11.1. NRgbImageGetPixel Function

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

```
NResult N_API NRgbImageGetPixel(
        HNImage hImage,
        NUInt x,
        NUInt y,
        NRgb * pValue
);
```

Parameters

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
pValue	[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	*hImage* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*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 npfRgb.

See Also

NRgbImage Module | HNImage | NRgb | NRgbImageSetPixel

4.2.11.2. 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

hImage	[in] Handle to image.

x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
pValue	[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	*hImage* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*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 `npfRgb`.

See Also

NRgbImage Module | HNImage | NRgb | `NRgbImageGetPixel`

4.2.12. 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 Pro Library

4.2.12.1. TiffLoadImageFromFile Function

Loads image from TIFF file.

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

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[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	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* 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

4.2.12.2. TiffLoadImageFromMemory Function

Loads image from memory buffer containing TIFF file.

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


Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[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	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* 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

4.2.13. Wsq Module

Provides functionality for loading and saving images in WSQ format.

Header file: Wsq.h.

Functions

WsqLoadImageFromFile	Loads image from WSQ file.
WsqLoadImageFromMemory	Loads image from memory buffer containing WSQ file.

WsqSaveImageToFile	Saves image to file in WSQ format with specified bitrate.
WsqSaveImageToMemory	Saves image to memory buffer in WSQ format with specified bitrate.

Macros

WSQ_DEFAULT_BIT_RATE	Specifies default bit rate (compression level).

See Also

NImages Pro Library

4.2.13.1. WsqLoadImageFromFile Function

Loads image from WSQ file.

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

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[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	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid.

Remarks

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

See Also

Wsq Module | HNImage | `WsqLoadImageFromMemory` | `WsqSaveImageToFile`

4.2.13.2. WsqLoadImageFromMemory Function

Loads image from memory buffer containing WSQ file.

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

Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[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	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid.

Remarks

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

See Also

Wsq Module | HNImage | `WsqLoadImageFromFile` | `WsqSaveImageToMemory`

4.2.13.3. WsqSaveImageToFile Function

Saves image to file in WSQ format with specified bitrate.

```
NResult N_API WsqSaveImageToFile(
        HNImage hImage,
        NFloat bitRate,
        const NChar * szFileName
);
```

Parameters

hImage	[in] Handle to image.
bitRate	[in] Specifies bit rate (compression level).
szFileName	[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	*hImage* or *szFileName* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*bitRate* is less than or equal to zero.

Remarks

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

bitRate can be set to WSQ_DEFAULT_BIT_RATE (0.75) that corresponds to 15:1 compression. The lower the bit rate is, the higher is the compression level and vice versa. Another common bit rate is 2.25 for 5:1 compression.

See Also

Wsq Module | HNImage | WSQ_DEFAULT_BIT_RATE | WsqSaveImageToMemory | WsqLoadImageFromFile

4.2.13.4. WsqSaveImageToMemory Function

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

```
NResult N_API WsqSaveImageToMemory(
        HNImage hImage,
        NFloat bitRate,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImage	[in] Handle to image.
bitRate	[in] Specifies bit rate (compression level).
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[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	*hImage*, *pBuffer* or *pBufferLength* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*bitRate* is less than or equal to zero.
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

Wsq Module | HNImage | WSQ_DEFAULT_BIT_RATE | `WsqSaveImageToFile` | `WsqLoadImageFromMemory`

Chapter 5. Reference (.NET)

This chapter contains reference of all libraries included in NImages Pro Add-On for .NET developers.

C# language is used where it is needed to provide sample code.

Libraries

Neurotec	Provides classes that provide infrastructure for Neurotechnologija components.
Neurotec.Images Pro	Provides classes that enable loading, saving and converting images in various formats.

5.1. Neurotec Library

Provides classes that provide infrastructure for Neurotechnologija components.

DLL: `Neurotec.dll`.

Namespaces

Neurotec	Contains classes that provide infrastructure for Neurotechnologija components.

5.1.1. Neurotec Namespace

Contains classes that provide infrastructure for Neurotechnologija components.

Classes

LicenceManagerException	The exception that is thrown when trying to register a Neurotechnologija library with License Manager server and an error has occurred.
NCore	This class supports internal Neurotechnologija libraries infrastructure and should not be used directly in your code.
NeurotecException	The exception that is thrown when unknown error occurred in one of Neurotechnologija libraries.

NotRegisteredException	The exception that is thrown when using unregistered Neurotechnologija library.
NParameters	This class supports internal Neurotechnologija libraries infrastructure and should not be used directly in your code.
NResult	This class supports internal Neurotechnologija libraries infrastructure and should not be used directly in your code.
ParameterException	The exception that is thrown when parameter code provided to a parameter value get or set method is not valid.
ParameterReadOnlyException	The exception that is thrown when parameter, which code is provided to a parameter value set method, is read-only.

Structures

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

Enumerations

NByteOrder	Specifies byte order.

5.1.1.1. NByteOrder Enumeration

Specifies byte order.

```
public enum NByteOrder
```

Members

Member	Description
BigEndian	Big-endian byte order.
LittleEndian	Little-endian byte order.

5.1.1.2. NIndexPair Structure

Represents pair of indexes.

Constructors

NIndexPair	Initializes a new instance of the NIndexPair structure.

Properties

Index1	Gets or sets first index of this NIndexPair.
Index2	Gets or sets second index of this NIndex-Pair.

5.1.1.2.1. Index1 Property

Gets or sets first index of this NIndexPair.

```
public int Index1 {get; set;}
```

Property value

First index of this NIndexPair.

5.1.1.2.2. Index2 Property

Gets or sets second index of this NIndexPair.

```
public int Index2 {get; set;}
```

Property value

Second index of this NIndexPair.

5.1.1.2.3. NIndexPair Constructor

```
public NIndexPair(
        int index1,
        int index2
);
```

Parameters

index1	First index of this NIndexPair.
index2	Second index of this NIndexPair.

5.1.1.3. NRational Structure

Represents a signed rational number.

Constructors

NRational	Initializes a new instance of the NRational structure.

Fields

Empty	Represents a NRational that is a null reference.

Properties

Denominator	Sets or retrieves the NRational value Denominator.
Numerator	Sets or retrieves the NRational value Numerator.

5.1.1.3.1. NRational Constructor

Initializes a new instance of the NRational structure.

```
public NRational(
        int numerator,
        int denominator
);
```

Parameters

numerator	Numerator of this NRational.
denominator	Denominator of this NRational.

5.1.1.3.2. Empty Field

Represents a NRational that is a null reference.

```
public static readonly NRational Empty
```

5.1.1.3.3. Denominator Property

Sets or retrieves the NRational value Denominator.

```
public int Denominator {get; set;}
```

Property value

Denominator of this NRational.

5.1.1.3.4. Numerator Property

Sets or retrieves the NRational value Numerator.

```
public int Numerator {get; set;}
```

Property value

Numerator of this NRational.

5.1.1.4. NURational Structure

Represents an unsigned rational number.

Constructors

NURational	Initializes a new instance of the NURational structure.

Fields

Empty	Represents a NURational that is a null reference.

Properties

Denominator	Sets or retrieves the NURational value Denominator.

Numerator	Sets or retrieves the NURational value Numerator.

5.1.1.4.1. NURational Constructor

Initializes a new instance of the NURational structure.

```
public NURational(
        int numerator,
        int denominator
);
```

Parameters

numerator	Numerator of this NURational.
denominator	Denominator of this NURational.

5.1.1.4.2. Empty Field

Represents a NURational that is a null reference.

```
public static readonly NURational Empty
```

5.1.1.4.3. Denominator Property

Sets or retrieves the NURational value Denominator.

```
public int Denominator {get; set;}
```

Property value

Denominator of this NURational.

5.1.1.4.4. Numerator Property

Sets or retrieves the NURational value Numerator.

```
public int Numerator {get; set;}
```

Property value

Numerator of this NURational.

5.1.1.5. NeurotecException Class

The exception that is thrown when unknown error occurred in one of Neurotechnologija libraries.

Properties

Code	Gets a error code.
Message	Gets a message that describes the current exception.

5.1.1.5.1. Code Property

Gets a error code.

```
public int Code {get;}
```

Property value

An error code.

5.1.1.5.2. Message Property

Gets a message that describes the current exception.

```
public override string Message {get;}
```

Property value

An error message.

5.2. Neurotec.Images Pro Library

Provides classes that enable loading, saving and converting images in various formats.

DLL: `Neurotec.Images.dll`.

Namespaces

Neurotec.Images	Contains classes that enable loading, saving and converting images in various formats.

5.2.1. Neurotec.Images Namespace

Contains classes that enable loading, saving and converting images in various formats.

Classes

Bmp	Provides functionality for loading and saving images in BMP format.
IHead	Provides functionality for loading and saving images in NIST IHead format.
Jpeg	Provides functionality for loading and saving images in JPEG format.
NGrayscaleImage	Provides functionality for managing 8-bit grayscale images.
NImage	Provides functionality for managing images.
NImageFile	Provides functionality for reading image files in format-neutral style.
NImageFormat	Provides functionality for loading and saving images in format-neutral style.
NImageFormat.ImageFormatCollection	Represents the collection of formats in a NImageFormat.
NImages	Provides library registration and other additional functionality.
NMonochromeImage	Provides functionality for managing 1-bit monochrome images.
NRgbImage	Provides functionality for managing 24-bit RGB images.
Tiff	Provides functionality for loading and saving images in TIFF format.
Wsq	Provides functionality for loading and saving images in WSQ format.

Structures

NPixelFormat	Provides functionality for working with pixel format.
NRgb	Represents an RGB color.

5.2.1.1. Bmp Class

Provides functionality for loading and saving images in BMP format.

Methods

LoadImage	Creates a new instance of the `NImage` class.
LoadImageFromBitmap	Creates a new instance of the `NImage` class from `Bitmap`.
LoadImageFromHBitmap	Creates a new instance of the `NImage` class from Windows HBITMAP.
SaveImage	Saves `NImage`.
SaveImageToBitmap	Saves `NImage` to `Bitmap`.
SaveImageToHBitmap	Saves `NImage` to Windows HBITMAP.

5.2.1.1.1. LoadImage Method.

Creates a new instance of the `NImage` class.

5.2.1.1.1.1. LoadImage (string)

Creates a new instance of the `NImage` class from file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

`fileName`	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.1.1.2. LoadImage (IntPtr, int)

Creates a new instance of the `NImage` class from memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.1.1.3. LoadImage (byte[])

Creates a new instance of the `NImage` class from byte array.

```
public static NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.1.2. LoadImageFromBitmap Method

Creates a new instance of the `NImage` class from `Bitmap`.

5.2.1.1.2.1. LoadImageFromBitmap (Bitmap)

Creates a new instance of the `NImage` class from `Bitmap`.

```
public static NImage LoadImageFromBitmap(
        Bitmap bitmap
);
```

Parameters

bitmap	A `Bitmap` class object.

Return Values

A NImage object.

See Also

NImage | SaveImageToBitmap

5.2.1.1.2.2. LoadImageFromBitmap (Bitmap, float, float)

Creates a new instance of the `NImage` class from `Bitmap` with specified resolution.

```
public static NImage LoadImageFromBitmap(
        Bitmap bitmap,
        float horzResolution,
        float vertResolution
);
```

Parameters

bitmap	A `Bitmap` class object.
horzResolution	A horizontal resolution in pixels per inch of fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.

Return Values

A NImage object.

See Also

NImage | SaveImageToBitmap

5.2.1.1.3. LoadImageFromHBitmap Method

Creates a new instance of the `NImage` class from Windows HBITMAP.

```
public static NImage LoadImageFromHBitmap(
        IntPtr hBitmap
);
```

Parameters

hBitmap	Pointer to handle that specifies Windows HBITMAP.

Return Values

A NImage object.

See Also

NImage | SaveImageToHBitmap

5.2.1.1.4. SaveImage Method

Saves `NImage`.

5.2.1.1.4.1. void SaveImage (NImage, string)

Saves `NImage` to file.

```
public static void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

NImage | LoadImage

5.2.1.1.4.2. void SaveImage (NImage, Stream)

Saves `NImage` to stream.

```
public static void SaveImage(
        NImage image,
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

NImage | LoadImage

5.2.1.1.4.3. byte[] SaveImage (NImage)

Saves `NImage` to byte array.

```
public static byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A byte array.

See Also

NImage | LoadImage

5.2.1.1.5. SaveImageToBitmap Method

Saves `NImage` to `Bitmap`.

```
public static Bitmap SaveImageToBitmap(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A `Bitmap` object.

See Also

NImage | LoadImageFromBitmap

5.2.1.1.6. SaveImageToHBitmap Method

Saves `NImage` to Windows HBITMAP.

```
public static IntPtr SaveImageToHBitmap(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A pointer to handle of Windows HBITMAP.

See Also

NImage | LoadImageFromHBitmap

5.2.1.2. IHead Class

Provides functionality for loading and saving images in NIST IHead format.

Methods

LoadImage	Creates `NImage` object from NIST IHead.
SaveImage	Saves `NImage` object in NIST IHead format.

5.2.1.2.1. LoadImage Method

Creates `NImage` object from NIST IHead.

5.2.1.2.1.1. LoadImage (string)

Creates `NImage` object from NIST IHead file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage

5.2.1.2.1.2. LoadImage (IntPtr, int)

Creates `NImage` object from NIST IHead memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage

5.2.1.2.1.3. LoadImage (byte[])

Creates `NImage` object from NIST IHead byte array.


```
public static NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage

5.2.1.2.2. SaveImage Method

Saves `NImage` object in NIST IHead format.

5.2.1.2.2.1. void SaveImage (NImage, string)

Saves `NImage` object to file in NIST IHead format.

```
public static void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

NImage

5.2.1.2.2.2. byte[] SaveImage (NImage)

Saves `NImage` object to byte array in NIST IHead format.

```
public static byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A array with saved image.

See Also

NImage

5.2.1.2.2.3. void SaveImage (NImage, Stream)

Saves `NImage` object to memory stream in NIST IHead format.

```
public static void SaveImage(
        NImage image,
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

NImage

5.2.1.3. Jpeg Class

Provides functionality for loading and saving images in JPEG format.

Methods

LoadImage	Creates NImage object.
SaveImage	Saves NImage object.

Constants

DefaultQuality	Specifies default JPEG quality.

5.2.1.3.1. LoadImage Method

Creates NImage object.

5.2.1.3.1.1. LoadImage (string)

Creates NImage object from JPEG file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.3.1.2. LoadImage (IntPtr, int)

Creates NImage object from memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.3.1.3. LoadImage (byte[])

Creates NImage object from byte array.

```
public static NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.3.2. SaveImage Method

Saves NImage object.

5.2.1.3.2.1. void SaveImage (NImage, string)

Saves NImage object to file in JPEG format.

```
public static void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

NImage | LoadImage | SaveImage

5.2.1.3.2.2. void SaveImage (NImage, int, string)

Saves NImage object to file in JPEG format with specified bit rate.


```
public static void SaveImage(
        NImage image,
        int quality,
        string fileName
);
```

Parameters

image	A NImage object.
quality	Specifies quality of JPEG image.
fileName	A string that contains the name of the file.

See Also

NImage | LoadImage | SaveImage

5.2.1.3.2.3. byte[] SaveImage (NImage)

Saves NImage object to byte array in JPEG format.

```
public static byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A byte array.

See Also

NImage | LoadImage | SaveImage

5.2.1.3.2.4. byte[] SaveImage (NImage, int)

Saves NImage object to byte array in JPEG format with specified bit rate.

```
public static byte[] SaveImage(
        NImage image,
        int quality
);
```


Parameters

image	A NImage object.
quality	Specifies quality of JPEG image.

Return Values

A byte array.

See Also

NImage | LoadImage | SaveImage

5.2.1.3.2.5. void SaveImage (NImage, Stream)

Saves NImage object to stream in JPEG format.

```
public static void SaveImage(
        NImage image,
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

NImage | LoadImage | SaveImage

5.2.1.3.2.6. void SaveImage (NImage, int, Stream)

Saves NImage object to stream in JPEG format with specified bit rate.

```
public static void SaveImage(
        NImage image,
        int quality,
        Stream stream
);
```

Parameters

image	A NImage object.

bitRate	Specifies quality of JPEG image.
stream	The data stream used to save the image.

See Also

NImage | LoadImage | SaveImage

5.2.1.4. NGrayscaleImage Class

Provides functionality for managing 8-bit grayscale images.

Properties

Item	Gets or sets the color of the specified pixel in NImage object.

5.2.1.4.1. NGrayscaleImage.Item Property

Gets or sets the color of the specified pixel in NImage object.

```
public const byte this[
        uint x,
        uint y
] {get; set;}
```

Parameters

x	The x coordinate of the pixel.
y	The y coordinate of the pixel.

Property value

A color of specified pixel.

5.2.1.5. NImage Class

Provides functionality for managing images.

Properties

Handle	Gets handle to unmanaged NImage object.

Height	Gets height of fingerprint image from NImage object.
HorzResolution	Gets horizontal resolution in pixels per inch of fingerprint image.
LongSize	Gets size of NImage object.
LongStride	Gets stride of fingerprint image from NImage object.
PixelFormat	Gets NPixelFormat of NImage object.
Pixels	Gets pointer to array of pixels from NImage object.
Size	Gets size of NImage object.
Stride	Gets stride of fingerprint image from NImage object.
VertResolution	Gets vertical resolution in pixels per inch of fingerprint image.
Width	Gets width of fingerprint image from NImage object.

Methods

Clone	Creates NImage object from another NImage object.
Create	Creates an empty NImage object.
Dispose	Releases the resources used by NImage.
FromBitmap	Creates `NImage` from Bitmap.
FromData	Creates `NImage` object from data.
FromFile	Creates `NImage` object from file.
FromHandle	Creates `NImage` object from handle.
FromHBitmap	Creates a new instance of the NImage class from Windows HBITMAP.
FromImage	Creates `NImage` object from another `NImage` object.
GetWrapper	Creates `NImage` object wrapper.

Save	Saves `NImage` object to file.
ToBitmap	Creates a `Bitmap`.
ToHBitmap	Creates Windows HBITMAP.

5.2.1.5.1. Handle Property

Gets handle to unmanaged NImage object.

```
public IntPtr Handle {get;}
```

Property value

A handle to unmanaged NImage object.

See Also

NImage

5.2.1.5.2. Height Property

Gets height of fingerprint image from NImage object.

```
public uint Height {get;}
```

Property value

A height of fingerprint image.

See Also

NImage Width | Stride

5.2.1.5.3. HorzResolution Property

Gets horizontal resolution in pixels per inch of fingerprint image.

```
public float HorzResolution {get;}
```

Property value

A horizontal resolution in pixels per inch of fingerprint image.

Remarks

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

See Also

NImage VertResolution

5.2.1.5.4. LongSize Property

Gets size of NImage object.

```
public ulong LongSize {get;}
```

Property value

A size of NImage object.

Remarks

Size of memory block containing image pixels is equal to image height multiplied by image stride. For more information see Height and Stride properties.

See Also

NImage | Height | Stride

5.2.1.5.5. LongStride Property

Gets stride of fingerprint image from NImage object.

```
public ulong LongStride {get;}
```

Property value

A stride of fingerprint image.

Stride (size of one row) of the image depends on image pixel format and width. It can not be less than value obtained with GetRowLongSize or GetRowSize methods with arguments obtained with PixelFormat and Width properties.

See Also

NImage

5.2.1.5.6. PixelFormat Property

Gets NPixelFormat of NImage object.

```
public NPixelFormat PixelFormat {get;}
```

Property value

A NPixelFormat structure.

See Also

NImage

5.2.1.5.7. Pixels Property

Gets pointer to array of pixels from NImage object.

```
public IntPtr Pixels {get;}
```

Property value

A pointer to pixel array.

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 PixelFormat, Width, Height, Stride, and Size properties.

See Also

NImage | PixelFormat | Width | Height | Stride | Size

5.2.1.5.8. Size Property

Gets size of NImage object.

```
public uint Size {get;}
```

Property value

A size of NImage object.

Remarks

Size of memory block containing image pixels is equal to image height multiplied by image stride. For more information see Height and Stride properties.

See Also

NImage | Height | Stride

5.2.1.5.9. Stride Property

Gets stride of fingerprint image from NImage object.

```
public uint Stride {get;}
```

Property value

A stride of fingerprint image.

Stride (size of one row) of the image depends on image pixel format and width. It can not be less than value obtained with GetRowLongSize or GetRowSize methods with arguments obtained with PixelFormat and Width properties.

See Also

NImage

5.2.1.5.10. VertResolution Property

Gets vertical resolution in pixels per inch of fingerprint image.

```
public float VertResolution {get;}
```

Property value

A vertical resolution in pixels per inch of fingerprint image.

Remarks

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

See Also

NImage | HorzResolution

5.2.1.5.11. Width Property

Gets width of fingerprint image from NImage object.

```
public uint Width {get;}
```

Property value

A width of fingerprint image.

See Also

NImage Height | Stride

5.2.1.5.12. Clone Method

Creates NImage object from another NImage object.

```
public object Clone();
```

Return Values

A NImage object.

See Also

NImage

5.2.1.5.13. Create Method

Creates an empty NImage object.

5.2.1.5.13.1. Create (NPixelFormat, uint, uint, uint, float, float)

Creates an empty NImage object.

```
public static NImage Create(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        uint stride,
        float horzResolution,
        float vertResolution
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution in pixels per inch of fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.

Return Values

A NImage object.

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride method.

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

See Also

NImage

5.2.1.5.13.2. Create (NPixelFormat, uint, uint, ulong, float, float)

Creates an empty NImage object.

```
public static NImage Create(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        ulong stride,
        float horzResolution,
        float vertResolution
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution in pixels per inch of fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.

Return Values

A NImage object.

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride method.

horzResolutionM and *vertResolution* can be zero if resolution is not applicable for

the image.

See Also

NImage

5.2.1.5.14. Dispose Method

Releases the resources used by NImage.

```
public void Dispose();
```

See Also

NImage

5.2.1.5.15. FromBitmap Method

Creates `NImage` from Bitmap.

```
public static NImage FromBitmap(
        Bitmap bitmap
);
```

Parameters

bitmap	An object used to work with images defined by pixel data.

Return Values

A NImage object.

See Also

NImage

5.2.1.5.16. FromData Method

Creates `NImage` object from data.

5.2.1.5.16.1. FromData (NPixelFormat, uint, uint, uint, float, float, uint, IntPtr)

Creates `NImage` object from data with specified resolution.

```
public static NImage FromData(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
```



```
        uint stride,
        float horzResolution,
        float vertResolution,
        uint srcStride,
        IntPtr srcPixels
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution in pixels per inch of fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.
srcStride	A stride of source fingerprint image.
srcPixels	A pointer to source pixel array.

Return Values

A NImage object.

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride property.

Format of memory block *srcPixels* points to must be the same as described in Pixels property, only stride is equal to *srcStride*.

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

See Also

NImage

5.2.1.5.16.2. FromData (NPixelFormat, uint, uint, ulong, float, float, ulong, IntPtr)

Creates NImage object from data with specified resolution.

```
public static NImage FromData(
```

```
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        ulong stride,
        float horzResolution,
        float vertResolution,
        ulong srcStride,
        IntPtr srcPixels
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution in pixels per inch of fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.
srcStride	A stride of source fingerprint image.
srcPixels	A pointer to source pixel array.

Return Values

A NImage object.

See Also

NImage

5.2.1.5.17. FromFile Method

Creates `NImage` object from file.

5.2.1.5.17.1. FromFile (string)

Creates `NImage` object from file.

```
public static NImage FromFile(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage

5.2.1.5.17.2. FromFile (string, NImageFormat)

Creates `NImage` object from file with specified NImageFormat.

```
public static NImage FromFile(
        string fileName,
        NPixelFormat imageFormat
);
```

Parameters

fileName	A string that contains the name of the file.
imageFormat	An image NImageFormat object.

Return Values

A NImage object.

See Also

NImage | NImageFormat

5.2.1.5.18. FromHandle Method

Creates `NImage` object from handle.

```
public static NImage FromHandle(
        IntPtr handle
);
```

Parameters

handle	A pointer to handle.

Return Values

A NImage object.

See Also

NImage | Handle

5.2.1.5.19. FromHBitmap Method

Creates a new instance of the NImage class from Windows HBITMAP.

```
public static NImage FromHBitmap(
        IntPtr hBitmap
);
```

Parameters

hBitmap	Pointer to handle that specifies Windows HBITMAP.

Return Values

A NImage object.

See Also

NImage

5.2.1.5.20. FromImage Method

Creates `NImage` object from another `NImage` object.

5.2.1.5.20.1. FromImage (NPixelFormat, uint, NImage)

Creates `NImage` object from another `NImage` object.

```
public static NImage FromImage(
        NPixelFormat pixelFormat,
        uint stride,
        NImage srcImage
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
stride	A stride of fingerprint image.

srcImage	A NImage source object.

Return Values

A NImage object.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride property.

See Also

NImage | NPixelFormat

5.2.1.5.20.2. FromImage (NPixelFormat, ulong, NImage)

Creates NImage object from another NImage object.

```
public static NImage FromImage(
        NPixelFormat pixelFormat,
        ulong stride,
        NImage srcImage
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
stride	A stride of fingerprint image.
srcImage	A NImage source object.

Return Values

A NImage object.

See Also

NImage | NPixelFormat

5.2.1.5.20.3. FromImage (NPixelFormat, uint, float, float, NImage)

Creates NImage object from another NImage object with specified resolution.

```
public static NImage FromImage(
        NPixelFormat pixelFormat,
```



```
        uint stride,
        float horzResolution,
        float vertResolution,
        NImage srcImage
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution in pixels per inch of fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.
srcImage	A `NImage` source object.

Return Values

A NImage object.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride property.

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

See Also

NImage | NPixelFormat

5.2.1.5.20.4. FromImage (NPixelFormat, ulong, float, float, NImage)

Creates `NImage` object from another `NImage` object with specified resolution.

```
public static NImage FromImage(
        NPixelFormat pixelFormat,
        ulong stride,
        float horzResolution,
        float vertResolution,
        NImage srcImage
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution in pixels per inch of fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.
srcImage	A `NImage` source object.

Return Values

A NImage object.

See Also

NImage | NPixelFormat

5.2.1.5.21. GetWrapper Method

Creates `NImage` object wrapper.

5.2.1.5.21.1. GetWrapper (NPixelFormat, uint, uint, uint, float, float, IntPtr, bool)

Creates `NImage` object wrapper.

```
public static NImage GetWrapper(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        uint stride,
        float horzResolution,
        float vertResolution,
        IntPtr pixels,
        bool ownsPixels
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution in pixels per inch of

	fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.
pixels	Pointer to memory block containing pixels for the image.
ownsPixels	Specifies whether pixels will be automatically deleted with the image (if set to true).

Return Values

A NImage object.

Remarks

For more information on image stride see Stride property.

Format of memory block pixels points to must be the same as described in Pixels property.

pixels must not be deleted during lifetime of the image. If *ownsPixels* is `true` 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 | NPixelFormat | Pixels

5.2.1.5.21.2. GetWrapper (NPixelFormat, uint, uint, ulong, float, float, IntPtr, bool)

Creates `NImage` object wrapper.

```
public static NImage GetWrapper(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        ulong stride,
        float horzResolution,
        float vertResolution,
        IntPtr pixels,
        bool ownsPixels
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution in pixels per inch of fingerprint image.
vertResolution	A vertical resolution in pixels per inch of fingerprint image.
pixels	Pointer to memory block containing pixels for the image.
ownsPixels	Specifies whether pixels will be automatically deleted with the image (if set to true).

Return Values

A NImage object.

Remarks

Format of memory block pixels points to must be the same as described in Pixels property.

See Also

NImage | NPixelFormat | Pixels

5.2.1.5.22. Save Method

Saves `NImage` object to file.

5.2.1.5.22.1. Save (string)

Saves `NImage` object to file.

```
public void Save(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

See Also

NImage

5.2.1.5.22.2. Save (string, NImageFormat)

Saves `NImage` object to file with specified NImageFormat.

```
public void Save(
        string fileName,
        NImageFormat imageFormat
);
```

Parameters

fileName	A string that contains the name of the file.
imageFormat	An image NImageFormat object.

See Also

NImage | NImageFormat

5.2.1.5.23. ToBitmap Method

Creates a `Bitmap`.

```
public Bitmap ToBitmap();
```

Return Values

A `Bitmap` object.

See Also

NImage

5.2.1.5.24. ToHBitmap Method

Creates Windows HBITMAP.

```
public IntPtr ToHBitmap();
```

Return Values

A Windows HBITMAP.

See Also

NImage

5.2.1.6. NImageFile Class

Provides functionality for reading image files in format-neutral style.

Properties

Format	Gets image format from `NImageFile` object.
IsOpened	Gets a value indicating whether the file is currently open.

Methods

Close	Closes a file which is associated with `NImageFile` object.
Dispose	Releases the resources used by `NImageFile`.
FromFile	Creates `NImageFile` object from file.
ReadImage	Reads image from `NImageFile` object.

5.2.1.6.1. Format Property

Gets image format from `NImageFile` object.

```
public virtual NImageFormat Format {get;}
```

Property value

A NImageFormat object.

5.2.1.6.2. IsOpened Property

Gets a value indicating whether the file is currently open.

```
public virtual bool IsOpened {get;}
```

Property value

`true` if file is open, `false` if file is closed.

See Also

Close

5.2.1.6.3. Close Method

Closes a file which is associated with `NImageFile` object.

```
public virtual void Close();
```

See Also

IsOpened

5.2.1.6.4. Dispose Method

Releases the resources used by `NImageFile`.

```
public void Dispose();
```

5.2.1.6.5. FromFile Method

Creates `NImageFile` object from file.

5.2.1.6.5.1. FromFile (string)

Creates `NImageFile` object from file.

```
public static NImageFile FromFile(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImageFile

See Also

A NImageFile

5.2.1.6.5.2. FromFile (string, NImageFormat)

Creates `NImageFile` object from file with specified image format.

```
public static NImageFile FromFile(
        string fileName,
        NImageFormat imageFormat
);
```

Parameters

fileName	A string that contains the name of the file.
imageFormat	An image NImageFormat object.

Return Values

A NImageFile

See Also

NImageFile | NImageFormat

5.2.1.6.6. ReadImage Method

Reads image from `NImageFile` object.

```
public virtual NImage ReadImage();
```

Return Values

A NImage object.

See Also

NImage

5.2.1.7. NImageFormat Class

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

Fields

Bmp	Specifies the BMP image format.
Formats	Specifies collection of supported image formats.

Gif	Specifies the GIF image format.
IHead	Specifies the NIST IHead image format.
Jpeg	Specifies the JPEG image format.
Png	Specifies the PNG image format.
Tiff	Specifies the TIFF image format.
Wsq	Specifies the WSQ image format.

Properties

CanRead	Gets a value indicating whether the current image format supports reading.
CanWrite	Gets a value indicating whether the current image format supports writing.
CanWriteMultiple	Gets a value indicating whether the current image format supports writing of multiple images.
DefaultFileExtension	Gets default file extension of the current image format.
FileFilter	Gets file filter of the current image format.
Name	Gets name of the current image format.

Methods

LoadImage	Loads `NImage`.
OpenFile	Opens `NImageFile`.
SaveImage	Saves `NImage`.
SaveImages	Saves array of `NImage`.
Select	Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.

5.2.1.7.1. Bmp Field


```
public static readonly NImageFormat Bmp;
```

5.2.1.7.2. Formats Field

```
public static readonly NImageFormat.ImageFormatCollection Formats;
```

5.2.1.7.3. Gif Field

```
public static readonly NImageFormat Gif;
```

5.2.1.7.4. IHead Field

```
public static readonly NImageFormat IHead;
```

5.2.1.7.5. Jpeg Field

```
public static readonly NImageFormat Jpeg;
```

5.2.1.7.6. Png Field

```
public static readonly NImageFormat Png;
```

5.2.1.7.7. Tiff Field

```
public static readonly NImageFormat Tiff;
```

5.2.1.7.8. Wsq Field

```
public static readonly NImageFormat Wsq;
```

5.2.1.7.9. CanRead Property

Gets a value indicating whether the current image format supports reading.

```
public virtual bool CanRead {get;}
```

Property value

`true` if image format can read, `false` if image format can not read.

See Also

CanWrite | CanWriteMultiple | Name | DefaultFileExtension | FileFilter

5.2.1.7.10. CanWrite Property

Gets a value indicating whether the current image format supports writing.

```
public virtual bool CanWrite {get;}
```

Property value

`true` if image format can write, `false` if image format can not write.

See Also

CanRead | CanWriteMultiple | Name | DefaultFileExtension | FileFilter

5.2.1.7.11. CanWriteMultiple Property

Gets a value indicating whether the current image format supports writing of multiple images.

```
public virtual bool CanWriteMultiple {get;}
```

Property value

`true` if image format can write the multiple images, `false` if image format can not write the multiple images.

See Also

CanRead | CanWrite | Name | DefaultFileExtension | FileFilter

5.2.1.7.12. DefaultFileExtension Property

Gets default file extension of the current image format.

```
public virtual string DefaultFileExtension {get;}
```

Property value

Default file extension.

See Also

CanRead | CanWrite | Name | DefaultFileExtension | FileFilter

5.2.1.7.13. FileFilter Property

Gets file filter of the current image format.

```
public virtual string FileFilter {get;}
```


Property value

An image format file filter

See Also

CanRead | CanWrite | Name | DefaultFileExtension | CanWriteMultiple

5.2.1.7.14. Name Property

Gets name of the current image format.

```
public virtual string Name {get;}
```

Property value

An image format name.

See Also

CanRead | CanWrite | FileFilter | DefaultFileExtension | CanWriteMultiple

5.2.1.7.15. LoadImage Method

Loads `NImage`.

5.2.1.7.15.1. LoadImage(string)

Loads `NImage` from file.

```
public virtual NImage LoadImage(
        string fileName
);
```

Parameters

`fileName`	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | SaveImage

5.2.1.7.15.2. LoadImage(IntPtr, int)

Loads `NImage` from memory buffer.

```
public virtual NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage | SaveImage

5.2.1.7.15.3. LoadImage(byte[])

Loads `NImage` from byte array.

```
public virtual NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | SaveImage

5.2.1.7.16. OpenFile Method

Opens `NImageFile`.

5.2.1.7.16.1. OpenFile(string)

Opens `NImageFile`.

```
public virtual NImageFile OpenFile(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImageFile object.

See Also

NImageFile | OpenFile

5.2.1.7.16.2. OpenFile(IntPtr, int)

Opens `NImageFile` from memory buffer.

```
public virtual NImageFile OpenFile(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImageFile object.

See Also

NImageFile | OpenFile

5.2.1.7.16.3. OpenFile(byte[])

Opens `NImageFile` from byte array.

```
public virtual NImageFile OpenFile(
        byte[] buffer
```



```
);
```

Parameters

buffer	A byte array.

Return Values

A NImageFile object.

See Also

NImageFile | OpenFile

5.2.1.7.17. SaveImage Method

Saves `NImage`.

5.2.1.7.17.1. void SaveImage (NImage, string)

Saves NImage to file.

```
public virtual void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

LoadImage | NImage

5.2.1.7.17.2. byte[] SaveImage(NImage)

Saves NImage to byte array.

```
public virtual byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A byte array.

See Also

LoadImage | NImage

5.2.1.7.17.3. void SaveImage(NImage, Stream)

Saves NImage to stream.

```
public virtual void SaveImage(
        NImage image,
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

LoadImage | NImage

5.2.1.7.18. SaveImages Method

Saves array of `NImage`.

5.2.1.7.18.1. void SaveImages (NImage[], string)

Saves array of NImage to file.

```
public virtual void SaveImages(
        NImage[] images,
        string fileName
);
```

Parameters

images	A NImage objects array.

fileName	A string that contains the name of the file.

See Also

LoadImage | NImage

5.2.1.7.18.2. byte[] SaveImages(NImage[])

Saves array of NImage to byte array.

```
public virtual byte[] SaveImages(
        NImage[] images
);
```

Parameters

images	A NImage objects array.

Return Values

A byte array.

See Also

LoadImage | NImage

5.2.1.7.18.3. void SaveImages(NImage[], Stream)

Saves array of NImage to stream.

```
public virtual void SaveImages(
        NImage[] images,
        Stream stream
);
```

Parameters

images	A NImage objects array.
stream	The data stream used to save the image.

See Also

LoadImage | NImage

5.2.1.7.19. Select Method

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

```
public static NImageFormat Select(
        string fileName,
        FileAccess fileAccess
);
```

Parameters

fileName	A string that contains the name of the file.
fileAccess	A file access.

Return Values

A NImageFormat object.

See Also

NImageFormat

5.2.1.8. NImageFormat.ImageFormatCollection Class

Represents the collection of formats in a NImageFormat.

Properties

Item	Gets the member from collection by index.

Methods

IndexOf	Returns the index within the collection of the specified image format.

5.2.1.8.1. ImageFormatCollection.Item Property

Gets the member from collection by index.

```
public NImageFormat this[
        int index
] {get;}
```


Parameters

x	The index of the element to get.

Property value

A NImageFormat object.

See Also

NImageFormat

5.2.1.8.2. IndexOf Method

Returns the index within the collection of the specified image format.

```
public int IndexOf(
        NImageFormat value
);
```

Parameters

value	A NImageFormat object.

Return Values

The zero-based index of the NImageFormat in the collection.

See Also

NImageFormat

5.2.1.9. NImages Class

Provides library registration and other additional functionality.

Properties

IsRegistered	Checks if Neurotec.NImages Pro library is registered.

Methods

GetGrayscaleColorWrapper	Creates NImage object wrapper.

Constants

DllName	Name of DLL containing unmanaged part of this class.

5.2.1.9.1. IsRegistered Property

Checks if Neurotec.NImages Pro library is registered.

```
public static bool IsRegistered {get;}
```

Property value

`true` if library is registered, `false` if library is not registered.

5.2.1.9.2. GetGrayscaleColorWrapper Method

Creates NImage object wrapper.

```
public static NImage GetGrayscaleColorWrapper(
        NImage image,
        NRgb minColor,
        NRgb maxColor
);
```

Parameters

image	A NImage object.
minColor	Specifies color to be used for black color.
maxColor	Specifies color to be used for white color.

Return Values

An NImage object.

Remarks

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

See Also

NImage

5.2.1.10. NMonochromeImage Class

Provides functionality for managing 1-bit monochrome images.

Properties

Item	Gets or sets the color of the specified pixel in NImage object.

Remarks

This class provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to Monochrome.

5.2.1.10.1. NMonochromeImage.Item Property

Gets or sets the color of the specified pixel in NImage object.

```
public bool this[
        uint x,
        uint y
] {get; set;}
```

Parameters

x	The x coordinate of the pixel.
y	The y coordinate of the pixel.

Property value

If pixel is black then gets/sets `false` and if it is white then gets/sets `true`.

See Also

NImage

5.2.1.11. NPixelFormat Struct

Provides functionality for working with pixel format.

Fields

Grayscale	Each pixel value is stored in 8 bits representing 256 shades of gray.

Monochrome	Each pixel value is stored in 1 bit representing either black or white color.
Rgb	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 these fields. However only these fields are provided for usage with this SDK.

Properties

BitsPerPixel	Gets number of bits used to store a pixel from NPixelFormat Fields.

Methods

CalcRowLongSize	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
CalcRowSize	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
Equals	Determines whether the specified Object is equal to the current Object.
GetHashCode	Is intended for a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.
GetRowLongSize	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.
GetRowSize	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.
IsValid	Checks whether current NPixelFormat value is valid.

5.2.1.11.1. Grayscale Field

Each pixel value is stored in 8 bits representing 256 shades of gray.

```
public static readonly NPixelFormat Grayscale;
```

See Also

NPixelFormat

5.2.1.11.2. Monochrome Field

Each pixel value is stored in 1 bit representing either black or white color.

```
public static readonly NPixelFormat Monochrome;
```

See Also

NPixelFormat

5.2.1.11.3. Rgb Field

Each pixel value is stored in 24 bits consisting of three 8-bit values representing red, green and blue color components.

```
public static readonly NPixelFormat Rgb;
```

See Also

NPixelFormat

5.2.1.11.4. BitsPerPixel Property

Gets number of bits used to store a pixel from NPixelFormat Fields.

```
public uint BitsPerPixel {get;}
```

Property value

A number of bits.

See Also

NPixelFormat Fields

5.2.1.11.5. CalcRowLongSize Methods

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.

5.2.1.11.5.1. CalcRowLongSize (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.

```
public static ulong CalcRowLongSize(
        uint bitCount,
        uint length
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified bits per pixel.

See Also

CalcRowLongSize

5.2.1.11.5.2. CalcRowLongSize (uint, uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel and alignment.

```
public static ulong CalcRowLongSize(
        uint bitCount,
        uint length,
        uint alignment
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified bits per pixel.

See Also

CalcRowLongSize

5.2.1.11.6. CalcRowSize Methods

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.

5.2.1.11.6.1. CalcRowSize (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.

```
public static uint CalcRowSize(
        uint bitCount,
        uint length
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified bits per pixel.

See Also

CalcRowSize

5.2.1.11.6.2. CalcRowSize (uint, uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel and alignment.

```
public static uint CalcRowSize(
        uint bitCount,
        uint length,
        uint alignment
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified bits per pixel.

See Also

CalcRowSize

5.2.1.11.7. Equals Method

Determines whether the specified Object is equal to the current Object.

```
public override bool Equals(
        object obj
);
```

Parameters

obj	The Object to compare with the current Object.

Return Values

`true` if the specified Object is equal to the current Object; otherwise, `false`.

See Also

NPixelFormat

5.2.1.11.8. GetHashCode Method

Is intended for a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.

```
public override int GetHashCode();
```

Return Values

A hash code for the current Object.

5.2.1.11.9. GetRowLongSize Method

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

5.2.1.11.9.1. GetRowLongSize (uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

```
public ulong GetRowLongSize(
        uint length
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

See Also

GetRowLongSize

5.2.1.11.9.2. GetRowLongSize (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat and alignment.

```
public ulong GetRowLongSize(
        uint length,
```

```
        uint alignment
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified NPixel-Format.

See Also

GetRowLongSize

5.2.1.11.10. GetRowSize Method

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

5.2.1.11.10.1. GetRowSize (uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

```
public uint GetRowSize(
        uint length
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified NPixel-Format.

See Also

GetRowSize

5.2.1.11.10.2. GetRowSize (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat and alignment.

```
public uint GetRowSize(
        uint length,
        uint alignment
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified NPixel-Format.

See Also

GetRowSize

5.2.1.11.11. IsValid Method

Checks whether current NPixelFormat value is valid.

```
public static bool IsValid(
        NPixelFormat value
);
```

Parameters

value	The NPixelFormat object.

Return Values

Returns `true` if the object is valid NPixelFormat, `false` if not.

5.2.1.12. NRgb Struct

Represents an RGB color.

Constructors

NRgb	Initializes a new instance of the NRgb structure.

Properties

Blue	Gets the blue component value of this NRGB structure.
Green	Gets the green component value of this NRGB structure.
Red	Gets the red component value of this NRGB structure.

5.2.1.12.1. NTgb constructor

Initializes a new instance of the NRgb structure.


```
public NRgb(
        byte red,
        byte green,
        byte blue
);
```

Parameters

red	The blue component value of this NRGB.
green	The green component value of this NRGB.
blue	The red component value of this NRGB.

5.2.1.12.2. Blue Property

Gets the blue component value of this NRGB structure.

```
public byte Blue {get; set;}
```

Property value

The blue component value of this NRGB.

5.2.1.12.3. Green Property

Gets the green component value of this NRGB structure.

```
public byte Green {get; set;}
```

Property value

The green component value of this NRGB.

5.2.1.12.4. Red Property

Gets the red component value of this NRGB structure.

```
public byte Red {get; set;}
```

Property value

The red component value of this NRGB.

5.2.1.13. NRgbImage Class

Provides functionality for managing 24-bit RGB images.

Properties

Item	Gets the pixel by index.

Remarks

This class provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to Rgb.

5.2.1.13.1. NRgbImage.Item Property

Gets the pixel by index.

```
public NRgb this[
        uint x,
        uint y
] {get; set;}
```

Property value

A NRgb structure.

Parameters

x	The x coordinate to get or set.
y	The y coordinate to get or set.

See Also

NRgb

5.2.1.14. Tiff Class

Provides functionality for loading and saving images in TIFF format.

Constructors

Tiff	Initializes a new instance of the `Tiff` class.

Methods

LoadImage	Creates NImage object.

5.2.1.14.1. Tiff Constructor

Initializes a new instance of the `Tiff`.

```
public Tiff();
```

See Also

`Tiff` Class

5.2.1.14.2. LoadImage Method

Creates NImage object.

5.2.1.14.2.1. LoadImage (string)

Creates NImage object from TIFF file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | LoadImage

5.2.1.14.2.2. LoadImage (IntPtr, int)

Creates NImage object from memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```


Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage | LoadImage

5.2.1.14.2.3. LoadImage (byte[])

Creates NImage object from byte array.

```
public static NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | LoadImage

5.2.1.15. Wsq Class

Provides functionality for loading and saving images in WSQ format.

Methods

LoadImage	Creates NImage object.
SaveImage	Saves NImage object.

Constants

DefaultBitRate	Specifies default bit rate (compression level).

5.2.1.15.1. LoadImage Method

Creates NImage object.

5.2.1.15.1.1. LoadImage (string)

Creates NImage object from WSQ file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.15.1.2. LoadImage (IntPtr, int)

Creates NImage object from memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.15.1.3. LoadImage (byte[])

Creates NImage object from byte array.

```
public static NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

5.2.1.15.2. SaveImage Method

Saves NImage object.

5.2.1.15.2.1. void SaveImage (NImage, string)

Saves NImage object to file in WSQ format.

```
public static void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

NImage | LoadImage | SaveImage

5.2.1.15.2.2. void SaveImage (NImage, float, string)

Saves NImage object to file in WSQ format with specified bit rate.

```
public static void SaveImage(
        NImage image,
        float bitRate,
        string fileName
);
```

Parameters

image	A NImage object.
bitRate	A bit rate (compression level).
fileName	A string that contains the name of the file.

Remarks

bitRate can be set to DefaultBitRate (0.75) that corresponds to 15:1 compression. The lower the bit rate is, the higher is the compression level and vice versa. Another common bit rate is 2.25 for 5:1 compression.

See Also

NImage | LoadImage | SaveImage

5.2.1.15.2.3. byte[] SaveImage (NImage)

Saves NImage object to byte array in WSQ format.

```
public static byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A byte array.

See Also

NImage | LoadImage | SaveImage

5.2.1.15.2.4. byte[] SaveImage (NImage, float)

Saves NImage object to byte array in WSQ format with specified bit rate.

```
public static byte[] SaveImage(
        NImage image,
        float bitRate
);
```

Parameters

image	A NImage object.
bitRate	A bit rate (compression level).

Return Values

A byte array.

Remarks

bitRate can be set to DefaultBitRate (0.75) that corresponds to 15:1 compression. The lower the bit rate is, the higher is the compression level and vice versa. Another common bit rate is 2.25 for 5:1 compression.

See Also

NImage | LoadImage | SaveImage

5.2.1.15.2.5. void SaveImage (NImage, Stream)

Saves NImage object to stream in WSQ format.

```
public static void SaveImage(
        NImage image,
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

NImage | LoadImage | SaveImage

5.2.1.15.2.6. void SaveImage (NImage, float, Stream)

Saves NImage object to stream in WSQ format with specified bit rate.

```
public static void SaveImage(
        NImage image,
        float bitRate,
        Stream stream
);
```

Parameters

image	A NImage object.
bitRate	A bit rate (compression level).
stream	The data stream used to save the image.

Remarks

bitRate can be set to DefaultBitRate (0.75) that corresponds to 15:1 compression. The lower the bit rate is, the higher is the compression level and vice versa. Another common bit rate is 2.25 for 5:1 compression.

See Also

NImage | LoadImage | SaveImage

Appendix A. Support

Neurotechnologija 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@neurotechnologija.com> for any help on solving the other development problems.

Appendix B. Change Log

This appendix lists NImages Pro Add-On 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 2.0.3.0

- UPD: NCore library to version 2.4.0.0.
- UPD: NImages Pro library to version 2.2.0.1.
- UPD: Neurotec library to version 2.4.0.0.
- UPD: Neurotec.Images Pro library to version 2.2.1.0.

Version 2.0.2.0

- UPD: NCore library to version 2.2.0.0.
- UPD: NImages Pro library to version 2.0.1.2.
- UPD: Neurotec library to version 2.2.0.0.
- UPD: Neurotec.Images Pro library to version 2.0.3.1.

Version 2.0.1.0

- UPD: Updated to be compatible with VeriFinger 5.0 SDK.
- UPD: NCore library to version 2.1.0.1.
- UPD: NImages Pro library to version 2.0.0.5.
- UPD: Neurotec library to version 2.1.0.0.
- UPD: Neurotec.Images Pro library to version 2.0.2.0.

Version 2.0.0.1

- UPD: NCore library to version 2.0.1.1.
- UPD: NImages Pro library to version 2.0.0.1.
- UPD: Neurotec library to version 2.0.1.1.

Version 2.0.0.0

Initial release. Contains the following components:

- NCore library version 2.0.1.0.
- NImages Pro library version 2.0.0.0.
- Neurotec library version 2.0.1.0.
- Neurotec.Images Pro library version 2.0.0.0.

B.1. Components

B.1.1. NCore Library

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.
- ADD: Stream integration with .NET.
- UPD: Exception integration with .NET.

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.

B.1.2. NImages Pro Library

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.
- FIX: Memory leak when reading WSQ files.

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

- UPD: Updated to use Microsoft Visual C++ Runtime Library 8.0.
- CHN: Renamed to `NImages` to be consistent with other components.

Version 2.0.0.2

- FIX: Some BMP files reading.

Version 2.0.0.1

- CHN: Minor internal changes.

Version 2.0.0.0

- CHN: Some interface changes.
- ADD: Support for monochrome and RGB images.
- ADD: Support for files with multiple images.
- ADD: BMP, TIFF (load-only) and NIST IHead formats support.
- FIX: WSQ reading errors with some files.

Version 1.0.0.2

- FIX: Minor fixes in header files.

Version 1.0.0.1

- FIX: Minor fixes.

Version 1.0.0.0

Initial release.

B.1.3. Neurotec Library

Version 2.4.0.0

- UPD: Reflects changes in unmanaged code.

Version 2.3.1.0

- ADD: NProcessorInfo class.

Version 2.3.0.0

- ADD: Stream integration with unmanaged code.
- UPD: Exception integration with unmanaged code.

Version 2.2.2.0

- ADD: More robust unmanaged error handling on Windows.

Version 2.2.1.0

- ADD: Classes for internal architecture support.

Version 2.2.0.0

- UPD: Updated to support changes in unmanaged code.

Version 2.1.0.0

- REM: LicenseManagerException class.
- CHN: Now uses Microsoft .NET Framework 2.0.

Version 2.0.1.2

- FIX: Minor fixes in parameters framework.

Version 2.0.1.1

- FIX: Minor fixes.
- UPD: Minor updates in structures.

Version 2.0.1.0

- ADD: NParameters class 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: All error codes are mapped to appropriate exceptions.

Version 1.0.0.0

Initial release.

B.1.4. Neurotec.Images Pro Library

Version 2.2.1.0

- CHN: NImageFile Close and Dispose methods behavior to be consistent with .NET Dispose pattern and removed IsOpened property in Pro version.

Version 2.2.0.0

- ADD: Loading from Stream.
- UPD: Saving to Stream.

Version 2.1.0.2

- UPD: Updated internal structure.

Version 2.1.0.1

- FIX: Object disposing issues.

Version 2.1.0.0

- UPD: Updated to support changes in unmanaged code.

Version 2.0.3.1

- FIX: Minor fixes.

Version 2.0.3.0

- UPD: Updated to support changes in unmanaged code.

Version 2.0.2.0

- ADD: NImages.GetOpenFileFilter, NImages.GetSaveFileFilter, NImages.GetOpenFileFilterString and NImages.GetSaveFileFilterString methods.
- FIX: Reading of read-only files.

Version 2.0.1.0

- ADD: NImage.FromHandle method overload with bool value specifying whether NImage will own the specified handle.
- CHN: Now uses Microsoft .NET Framework 2.0.

Version 2.0.0.0

- CHN: Assembly renamed to `Neurotec.Images.dll` to be consistent with other components.
- CHN: Some interface changes.
- ADD: Support for monochrome and RGB images.
- ADD: Support for files with multiple images.
- ADD: NIST IHead format support.
- ADD: BMP and TIFF (load-only) low-level support.

Version 1.0.0.0

Initial release.

