Template Management and Conversion 1.1 Add-On for VeriFinger/FingerCell

Template Management and Conversion 1.1 Add-On for VeriFinger/FingerCell

Published April 29, 2006. Version 1.1.0.0
Copyright © 2005-2006 Neurotechnologija

Table of Contents

Copyright © 2006 Neurotechnologija

Copyright © 2006 Neurotechnologija

Copyright © 2006 Neurotechnologija

Copyright © 2006 Neurotechnologija

Copyright © 2006 Neurotechnologija

Copyright © 2006 Neurotechnologija

Copyright © 2006 Neurotechnologija

Copyright © 2006 Neurotechnologija

Copyright © 2006 Neurotechnologija

List of Figures

List of Tables

Chapter 1. About

Template Management and Conversion Add-On allows to integrate the support for fingerprint template standards to existing biometric system based on VeriFinger SDK or FingerCell EDK. Thus, the system will be able to perform template conversion from and to ANSI/INCITS 378-2004 (Finger Minutiae Format for Data Interchange) and ANSI/NIST-ITL 1-2000 (Data Format for the Interchange of Fingerprint, Facial, & Scar Mark & Tattoo (SMT) Information), as well as conversion between VeriFinger, FingerCell and MegaMatcher fingerprint template formats.

1.1. Platforms Supported

Template Management and Conversion Add-On supports platforms based on x86 processor architecture. Libraries for Windows and Linux operating systems are provided. .NET wrappers of Windows libraries are provided for .NET developers.

1.2. System Requirements

System requirements for installation and usage of components for Microsoft Windows OS:

- PC with x86 compatible CPU
- Windows 9x/ME/2000/XP/2003 OS
- .NET framework 1.1 (for .NET components only)
- Microsoft Visual Studio .Net 2003

System requirements for installation and usage of components for Linux OS:

- Computer with x86 compatible CPU
- Linux OS 2.4 or newer
- GCC-3.3.x or newer
- GNU Make 3.80 or newer

1.3. Licensing

Template Management and Conversion Add-On has a shared license. This means that a license for Template Management and Conversion Add-On should be obtained only once.

Components of the SDK are not additionally licensed (they can be freely distributed in binary form with developed application or system based on MegaMatcher SDK or VeriFinger/FingerCell SDK).

Chapter 2. Overview

The Template Management and Conversion Add-On can be divided into the following parts:

- Template Management. Enables reading, writing and editing MegaMatcher and VeriFinger/FingerCell templates.
- Template Conversion. Enables conversion of MegaMatcher and VeriFinger/FingerCell templates to and from standard templates.

2.1. Template Management

Template (or biometric template) is a compact digital representation of biometric characteristics such as fingerprint, face, voice, etc. This product currently supports fingerprint templates only.

Fingerprint templates that are used in this product are typically extracted from fingerprint images using MegaMatcher Extractor library, VeriFinger library or FingerCell library. This type of template is called Neurotechnologija Fingerprint Record (NFRecord). There are two versions of NFRecord, namely version 1.0 and version 2.0. Version 1.0 NFRecord is used in VeriFinger and FingerCell and Version 2.0 – in MegaMatcher. Both versions store G (average fingerprint ridge density), fingerprint minutiae and optionally singular points and blocked orientation.

Each minutia is described by location (x and y coordinates in 500 dpi units), angle (in pi/128 units) and type (line end, line bifurcation or unknown). Also minutia can have additional (optional) information, namely quality (2.0 only), curvature (curvature level of ridges near minutia), G (density level of ridges near minutia) and ridge counts to neighbouring minutiae (2.0 only).

Each singular point is described by location (x and y coordinates in 500 dpi units) and type (core, delta or double core). Core points can have angle and delta points can have 3 angles (2.0 only). These angles are in pi/128 units and are -1 if absent.

Blocked orientations are obsolete and used for compatibility with VeriFinger 4.1 and FingerCell 1.1 only.

NFRecord 2.0 stores also quality (fingerprint quality level) and information about finger (finger position, fingerprint pattern class), fingerprint capture (impression type) and fingerprint image (size and resolution).

Differences between NFRecord versions are summarized in Table 2.1, “Information stored in different versions of NFRecord” and their usage in Table 2.2, “Optional NFRecord information usage in Neurotechnologija products”.

Table 2.1. Information stored in different versions of NFRecord

Can store \ NFRecord version	1.0	2.0
G	Yes	Yes
Quality	No	Yes
Information about finger, fingerprint capture and fingerprint image	No	Yes
Minutia quality	No	Yes
Minutia curvature	Yes	Yes
Minutia G	Yes	Yes
Ridge counts to neighbouring minutiae	No	Yes
Singular points	Yes (delta points have no angles)	Yes
Blocked orientations	Yes	Yes (for compatibility)

Table 2.2. Optional NFRecord information usage in Neurotechnologija products

Usage of \ Product	VeriFinger 4.1, FingerCell 1.1	VeriFinger 4.2, FingerCell 1.2	MegaMatcher 1.0
NFRecord version	1.0	1.0	2.0
Minutia quality	No	No	No
Minutiae curvature	Optional	Optional	Optional
Minutia G	No	Optional	Optional
Ridge counts to neighbouring minutiae	No	No	Optional
Blocked orientations	Optional	Optional, for compatibility	No (Optional if VeriFinger extractor/ matcher is used)

Copyright © 2006 Neurotechnologija

NFRecord is persisted in memory block (byte array) which can be stored in database or file or sent to another computer via network. This persisted state of NFRecord is called packed NFRecord and it is the piece of information witch extractors of VeriFinger, FingerCell and MegaMatcher return and matchers of VeriFinger and FingerCell should receive. Information from packed NFRecord can be retrieved, edited and packed NFRecord can be created with the help of NFRecord module (Neurotec.Biometrics.NFRecord class in .NET).

A collection of NFRecords (since NFRecord version 2.0) can be stored in Neurotechnologija Fingerprint Template (NFTemplate) to consolidate information about all person's fingerprints. In the same way as NFRecord, NFTemplate is persisted in memory block (packed NFTemplate). NFRecords can be retrieved from packed NFTemplate and packed NFTemplate can be created from NFRecords with the help of NFTemplate module (Neurotec.Biometrics.NFTemplate class in .NET).

NFTemplate (and other biometric templates in future versions) can be stored in Neurotechnologija Template (NTemplate) to consolidate information about all person's biometric characteristics. In the same way as NFTemplate, NTemplate is persisted in memory block (packed NTemplate) and it is a piece of MegaMatcher Matcher information should receive. NFTemplate can be retrieved from packed NTemplate and packed NTemplate can be created from NFTemplate with the help of NTemplate module (Neurotec.Biometrics.NTemplate class in .NET).

Figure 2.1, “Structure of template.” shows a brief structure of templates.

Copyright © 2006 Neurotechnologija

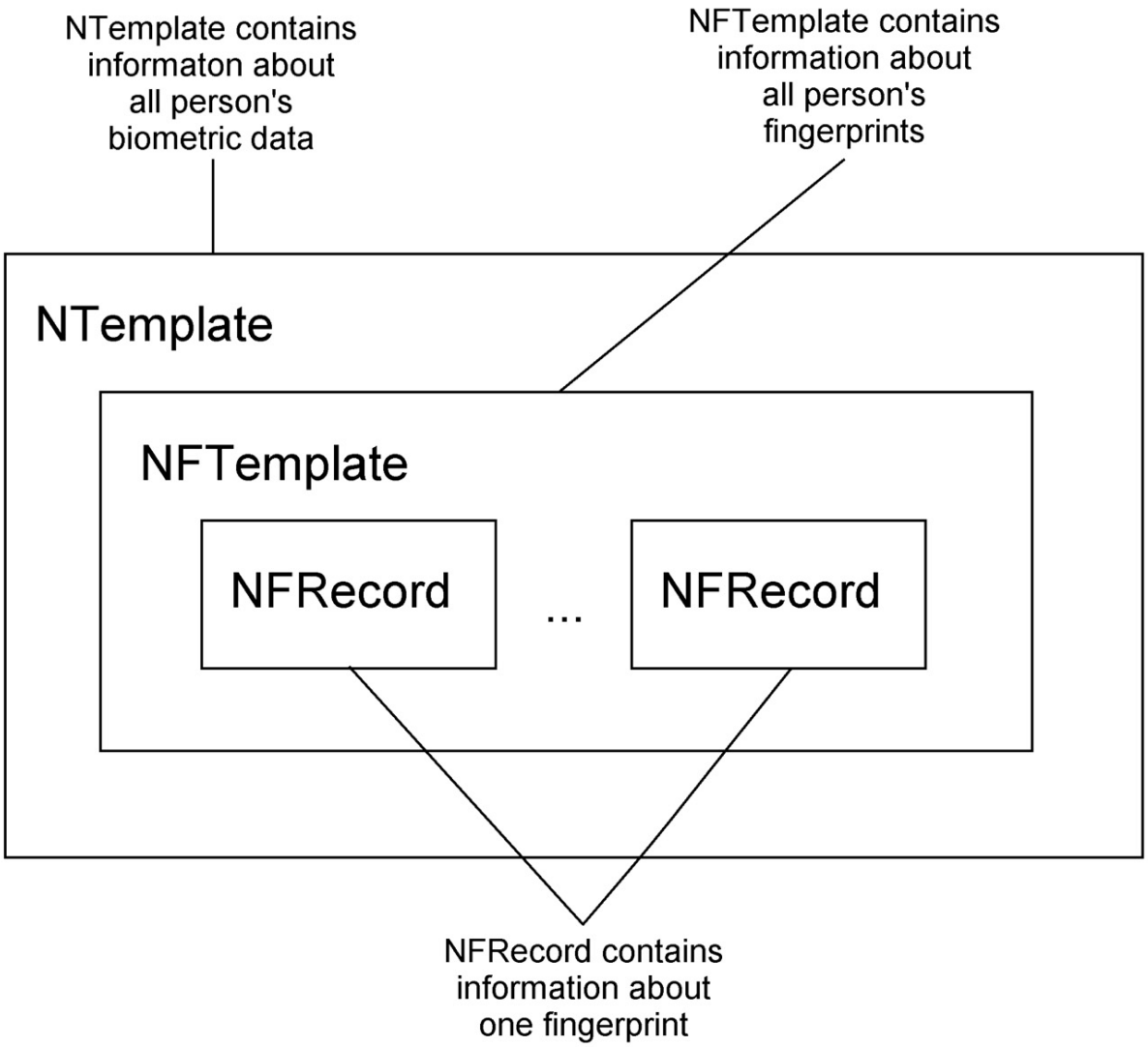

Figure 2.1. Structure of template.

2.2. Template Conversion

NTemplates, NFTemplates and NFRecords (see Template Management) can be converted to and/or from:

- Finger Minutiae Records (FMRecord, Finger Minutiae Format for Data Interchange ANSI/INCITS 378-2004 standard) with the help of FMRecord module (Neurotec.Biometrics.FMRecord class in .NET). This module (class in .NET) also enables reading, writing and editing of FMRecords.

Copyright © 2006 Neurotechnologija

- ANSI/NIST Files (ANTemplate, Data Format for the Interchange of Fingerprint, Facial, & Scar Mark & Tattoo (SMT) Information ANSI/NIST-ITL 1-2000 standard) with the help of ANTemplate module (Neurotec.Biometrics.ANTemplate class in .NET).

Chapter 3. Using

3.1. Template Management

This section contains template conversion examples including (but not limiting to) following:

- Creating MegaMatcher templates for matcher in Section 3.1.1, “Packing NFRecord to NTemplate”.
- Converting VeriFinger/FingerCell templates to MegaMatcher templates in Section 3.1.3, “Converting Packed NFRecord from Version 1.0 Format to Current (Version 2.0) Format” and Section 3.1.4, “Packing Version 1.0 Format NFRecord to NTemplate”.
- Converting MegaMatcher templates to VeriFinger/FingerCell templates in Section 3.1.5, “Converting Packed NFRecord from Current (Version 2.0) Format to Version 1.0 Format” and Section 3.1.6, “Retrieving Packed NFRecord in Version 1.0 Format from Packed NTemplate”.

3.1.1. Packing NFRecord to NTemplate

High-Level Example (Simpler)

```
[C]

#include "NTemplate.h"

void PackedNFRecordToPackedNTemplate1(
        NSizeType nfRecordSize, // size of packed NFRecord
        const void * nfRecord // pointer to memory block
                 // that contains packed NFRecord
)
{
        HNTemplate hNTemplate; // handle to NTemplate object
        HNFTemplate hNFTemplate; // handle to NFTemplate object
        HNFRecord hNFRecord; // handle to NFRecord object
        NSizeType nTemplateSize; // size of packed NTemplate
        void * nTemplate; // pointer to memory block
                // that contains packed NTemplate

        // Create new NTemplate object
        NTemplateCreate(&hNTemplate);
        // Add new NFTemplate to NTemplate
        NTemplateAddFingers(hNTemplate, &hNFTemplate);
        // Unpack packed NFRecord to newly created object and add it to NFTemplate
        NFTemplateAddRecordFromMemory(hNFTemplate, nfRecord, nfRecordSize,
                0, &hNFRecord);
        // Calculate size of packed NTemplate
        NTemplateGetSize(hNTemplate, 0, &nTemplateSize);
        // Allocate memory for packed NTemplate
        nTemplate = NAlloc(nTemplateSize);
        // Pack NTemplate
        NTemplateSaveToMemory(hNTemplate, nTemplate, nTemplateSize,
```

```
                0, &nTemplateSize);
        // ...
        // Do something with packed NTemplate
        // ...
        // Free memory allocated for packed NTemplate
        NFree(nTemplate);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateManagement
{
        void PackedNFRecordToPackedNTemplate1(
                byte[] packedNFRecord // packed NFRecord
        )
        {
                // Create new NTemplate object
                NTemplate nTemplate = new NTemplate();
                // Add new NFTemplate to NTemplate
                NFTemplate nfTemplate = nTemplate.AddFingers();
                // Unpack packed NFRecord to newly created object
                // and add it to NFTemplate
                NFRecord nfRecord = nfTemplate.Records.Add(packedNFRecord);
                // Pack NTemplate
                byte[] packedNTemplate = nTemplate.Save();
                // ...
                // Do something with packed NTemplate
                // ...
        }
}
```

Low-Level Example (Faster and Less Memory Consuming)

```
[C]

#include "NTemplate.h"

void PackedNFRecordToPackedNTemplate2(
        NSizeType nfRecordSize, // size of packed NFRecord
        const void * nfRecord // pointer to memory block
                 // that contains packed NFRecord
)
{
        NSizeType nfTemplateSize; // size of packed NFTemplate
        void * nfTemplate; // pointer to memory block
                // that contains packed NFTemplate
        NSizeType nTemplateSize; // size of packed NTemplate
        void * nTemplate; // pointer to memory block
                // that contains packed NTemplate
```

```
        // Calculate size of packed NFTemplate with one given NFRecord
        NFTemplateCalculateSize(1, &nfRecordSize, &nfTemplateSize);
        // Allocate memory for packed NFTemplate
        nfTemplate = NAlloc(nfTemplateSize);
        // Pack one given NFRecord to NFTemplate
        NFTemplatePack(1, &nfRecord, &nfRecordSize, nfTemplate, nfTemplateSize,
                &nfTemplateSize);
        // Calculate size of packed NTemplate with NFTemplate
        NTemplateCalculateSize(nfTemplateSize, &nTemplateSize);
        // Allocate memory for packed NTemplate
        nTemplate = NAlloc(nTemplateSize);
        // Pack NFTemplate to NTemplate
        NTemplatePack(nfTemplate, nfTemplateSize, nTemplate, nTemplateSize,
                &nTemplateSize);
        // Free memory allocated for packed NFTemplate
        NFree(nfTemplate);
        // ...
        // Do something with packed NTemplate
        // ...
        // Free memory allocated for packed NTemplate
        NFree(nTemplate);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateManagement
{
        void PackedNFRecordToPackedNTemplate2(
                byte[] packedNFRecord // packed NFRecord
        )
        {
                // Pack the given NFRecord to NFTemplate
                byte[] packedNFTemplate = NFTemplate.Pack(packedNFRecord);
                // Pack NFTemplate to NTemplate
                byte[] packedNTemplate = NTemplate.Pack(packedNFTemplate);
                // ...
                // Do something with packed NTemplate
                // ...
        }
}
```

3.1.2. Retrieving Packed NFRecord from Packed NTemplate

High-Level Example (Simpler)

```
[C]

#include "NTemplate.h"

void PackedNFRecordFromPackedNTemplate1(
```

```
        NSizeType nTemplateSize, // size of packed NTemplate
        const void * nTemplate // pointer to memory block
                // that contains packed NTemplate
)
{
        HNTemplate hNTemplate; // handle to NTemplate object
        HNFTemplate hNFTemplate; // handle to NFTemplate object
        HNFRecord hNFRecord; // handle to NFRecord object
        NSizeType nfRecordSize; // size of packed NFRecord
        void * nfRecord; // pointer to memory block
                // that contains packed NFRecord

        // Create NTemplate object from packed NTemplate
        NTemplateCreateFromMemory(nTemplate, nTemplateSize, 0, NULL, &hNTemplate);
        // Retrieve NFTemplate object from NTemplate object
        NTemplateGetFingers(hNTemplate, &hNFTemplate);
        // Retrieve first NFRecord object from NFTemplate object;
        NFTemplateGetRecord(hNFTemplate, 0, &hNFRecord);
        // Calculate packed size of NFRecord object
        NFRecordGetSize(hNFRecord, 0, &nfRecordSize);
        // Allocate memory for packed NFRecord
        nfRecord = NAlloc(nfRecordSize);
        // Pack NFRecord object
        NFRecordSaveToMemory(hNFRecord, nfRecord, nfRecordSize, 0, &nfRecordSize);
        // Delete NTemplate object
        NTemplateFree(hNTemplate);
        // ...
        // Do something with packed NFRecord
        // ...
        // Free memory allocated for packed NFRecord
        NFree(nfRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateManagement
{
        void PackedNFRecordFromPackedNTemplate1(
                byte[] packedNTemplate // packed NTemplate
        )
        {
                // Create NTemplate object from packed NTemplate
                NTemplate nTemplate = new NTemplate(packedNTemplate);
                // Retrieve NFTemplate object from NTemplate object
                NFTemplate nfTemplate = nTemplate.Fingers;
                // Retrieve first NFRecord object from NFTemplate object;
                NFRecord nfRecord = nfTemplate.Records[0];
                // Pack NFRecord object
                byte[] packedNFRecord = nfRecord.Save();
                // ...
                // Do something with packed NFRecord
                // ...
```

```
        }
}
```

Low-Level Example (Faster and Less Memory Consuming)

```
[C]

#include "NTemplate.h"

void PackedNFRecordFromPackedNTemplate2(
        NSizeType nTemplateSize, // size of packed NTemplate
        const void * nTemplate // pointer to memory block
                // that contains packed NTemplate
)
{
        NSizeType nfTemplateSize; // size of packed NFTemplate
        const void * nfTemplate; // pointer to packed NFTemplate
        NInt recordCount; // record count in NFTemplate
        const void * * arNFRecords; // pointer to array of
                // pointers to packed NFRecords
        NSizeType * arNFRecordSizes; // pointer to array of
                // sizes of packed NFRecords
        NSizeType nfRecordSize; // size of packed NFRecord
        const void * nfRecord; // pointer to packed NFRecord

        // Unpack packed NFTemplate from packed NTemplate
        NTemplateUnpack(nTemplate, nTemplateSize, NULL, NULL, NULL, NULL,
                &nfTemplate, &nfTemplateSize);
        // Unpack record count from packed NFTemplate
        NFTemplateUnpack(nfTemplate, nfTemplateSize, NULL, NULL, NULL, NULL,
                &recordCount, NULL, NULL);
        // Allocate array of pointers to packed NFRecords
        // and array of sizes of packed NFRecords
        arNFRecords = (const void * *)NAlloc(recordCount * sizeof(const void *));
        arNFRecordSizes = (NSizeType *)NAlloc(recordCount * sizeof(NSizeType));
        // Unpack packed NFRecords from packed NFTemplate
        NFTemplateUnpack(nfTemplate, nfTemplateSize, NULL, NULL, NULL, NULL,
                NULL, arNFRecords, arNFRecordSizes);
        // Retrieve first packed NFRecord
        nfRecord = arNFRecords[0];
        nfRecordSize = arNFRecordSizes[0];
        // Free array of pointers to packed NFRecords
        // and array of sizes of packed NFRecords
        NFree(arNFRecords);
        NFree(arNFRecordSizes);
        // ...
        // Do something with packed NFRecord
        // ...
}
```

3.1.3. Converting Packed NFRecord from Version 1.0 Format to Current (Version 2.0) Format

```
[C]

#include "NFRecordV1.h"

void PackedNFRecordV1ToPackedNFRecord(
        NSizeType nfRecordV1Size, // size of packed NFRecord in version 1.0 format
        const void * nfRecordV1 // pointer to memory block that contains
                // packed NFRecord in version 1.0 format
)
{
        HNFRecord hNFRecord; // handle to NFRecord object
        NSizeType nfRecordSize; // size of packed NFRecord
        void * nfRecord; // pointer to memory block that contains packed NFRecord

        // Create NFRecord object from packed NFRecord in version 1.0 format
        NFRecordCreateFromMemory(nfRecordV1, nfRecordV1Size, 0, NULL, &hNFRecord);
        // Calculate packed size of NFRecord object
        NFRecordGetSize(hNFRecord, 0, &nfRecordSize);
        // Allocate memory for packed NFRecord
        nfRecord = NAlloc(nfRecordSize);
        // Pack NFRecord object
        NFRecordSaveToMemory(hNFRecord, nfRecord, nfRecordSize, 0, &nfRecordSize);
        // Delete NFRecord object
        NFRecordFree(hNFRecord);
        // ...
        // Do something with packed NFRecord
        // ...
        // Free memory allocated for packed NFRecord
        NFree(nfRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateManagement
{
        public void PackedNFRecordV1ToPackedNFRecord(
                byte[] packedNFRecordV1 // packed NFRecord in version 1.0 format
        )
        {
                // Create NFRecord object from packed NFRecord in version 1.0 format
                NFRecord nfRecord = new NFRecord(packedNFRecordV1);
                // Pack NFRecord object
                byte[] packedNFRecord = nfRecord.Save();
                // ...
                // Do something with packed NFRecord
                // ...
        }
}
```

3.1.4. Packing Version 1.0 Format NFRecord to NTemplate

To pack version 1.0 NFRecord to NTemplate it has first be converted to current (version 2.0) format. See Section 3.1.3, “Converting Packed NFRecord from Version 1.0 Format to Current (Version 2.0) Format” and Section 3.1.1, “Packing NFRecord to NTemplate”.

3.1.5. Converting Packed NFRecord from Current (Version 2.0) Format to Version 1.0 Format

```
[C]

#include "NFRecordV1.h"

void PackedNFRecordToPackedNFRecordV1(
        NSizeType nfRecordSize, // size of packed NFRecord
        const void * nfRecord // pointer to memory block that contains packed NFRecord
)
{
        HNFRecord hNFRecord; // handle to NFRecord object
        NUInt flags = 0; // flags that control packing in version 1.0 format;
                // specify NFR_SKIP_GS | NFR_SAVE_BLOCKED_ORIENTS
                // instead of 0 for VeriFinger 4.1 / FingerCell 1.1
        NSizeType nfRecordV1Size; // size of packed NFRecord in version 1.0 format
        void * nfRecordV1; // pointer to memory block that contains
                // packed NFRecord in version 1.0 format

        // Create NFRecord object from packed NFRecord
        NFRecordCreateFromMemory(nfRecord, nfRecordSize, 0, NULL, &hNFRecord);
        // Calculate packed size of NFRecord object in version 1.0 format
        NFRecordGetSizeV1(hNFRecord, flags, &nfRecordV1Size);
        // Allocate memory for packed NFRecord in version 1.0 format
        nfRecordV1 = NAlloc(nfRecordV1Size);
        // Pack NFRecord object in version 1.0 format
        NFRecordSaveToMemoryV1(hNFRecord, nfRecordV1, nfRecordV1Size,
                flags, &nfRecordSize);
        // Delete NFRecord object
        NFRecordFree(hNFRecord);
        // ...
        // Do something with packed NFRecord in version 1.0 format
        // ...
        // Free memory allocated for packed NFRecord in version 1.0 format
        NFree(nfRecordV1);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateManagement
{
        public void PackedNFRecordToPackedNFRecordV1(
                byte[] packedNFRecord // packed NFRecord
        )
        {
```

```
                uint flags = 0; // flags that control packing in version 1.0 format;
                        // specify NFRecord.FlagSkipGs
                        // | NFRecord.FlagSaveBlockedOrients
                        // instead of 0 for VeriFinger 4.1 / FingerCell 1.1
                // Create NFRecord object from packed NFRecord
                NFRecord nfRecord = new NFRecord(packedNFRecord);
                // Pack NFRecord object in version 1.0 format
                byte[] packedNFRecordV1 = nfRecord.SaveV1(flags);
                // ...
                // Do something with packed NFRecord in version 1.0 format
                // ...
        }
}
```

3.1.6. Retrieving Packed NFRecord in Version 1.0 Format from Packed NTemplate

To retrieve version 1.0 NFRecord from NTemplate, NFRecord in current (version 2.0) format has first to be retrieved from NTemplate and then converted to version 1.0 format. See Section 3.1.2, “Retrieving Packed NFRecord from Packed NTemplate” and Section 3.1.5, “Converting Packed NFRecord from Current (Version 2.0) Format to Version 1.0 Format”.

3.2. Template Conversion

This section contains template conversion examples including (but not limiting to) following:

- Converting MegaMatcher templates to ANSI/INCITS 378-2004 standard templates in Section 3.2.4, “Converting NTemplate to FMRecord” and Section 3.2.1, “Converting NFRecord to FMRecord”.
- Converting VeriFinger/FingerCell templates to ANSI/INCITS 378-2004 standard templates in Section 3.2.2, “Converting Version 1.0 Format NFRecord to FMRecord”.
- Converting two-finger VeriFinger/FingerCell templates to ANSI/INCITS 378-2004 standard templates in Section 3.2.3, “Converting Two Version 1.0 Format NFRecords to FMRecord”.
- Converting ANSI/INCITS 378-2004 standard templates to MegaMatcher templates in Section 3.2.7, “Converting FMRecord to NTemplate” and Section 3.2.5, “Converting FMRecord to NFRecord”.
- Converting ANSI/INCITS 378-2004 standard templates to VeriFinger/FingerCell templates in Section 3.2.5, “Converting FMRecord to NFRecord”.
- Converting two-finger ANSI/INCITS 378-2004 standard templates to VeriFinger/FingerCell templates in Section 3.2.6, “Converting Two-Finger FMRecord to NFRecords in Version 1.0 Format”.
- Converting MegaMatcher templates to ANSI/NIST-ITL 1-2000 standard files in Section 3.2.10, “Converting NTemplate to ANTemplate” and Section 3.2.8, “Converting NFRecord to ANTemplate”.
- Converting VeriFinger/FingerCell templates to ANSI/NIST-ITL 1-2000 standard files in Section 3.2.9, “Converting Version 1.0 Format NFRecord to ANTemplate”.
- Converting ANSI/NIST-ITL 1-2000 standard files to MegaMatcher templates in Section 3.2.12, “Converting ANTemplate to NTemplate” and Section 3.2.11, “Converting

ANTemplate to NFRecord".
- Converting ANSI/NIST-ITL 1-2000 standard files to VeriFinger/FingerCell templates in Section 3.2.11, "Converting ANTemplate to NFRecord".

3.2.1. Converting NFRecord to FMRecord

```
[C]

#include "FMRecord.h"

void PackedNFRecordToStoredFMRecord(
        NSizeType nfRecordSize, // size of packed NFRecord
        const void * nfRecord // pointer to memory block
                // that contains packed NFRecord
)
{
        HNFRecord hNFRecord; // handle to NFRecord object
        HFMRecord hFMRecord; // handle to FMRecord object
        NSizeType fmRecordSize; // size of FMRecord
        void * fmRecord; // pointer to memory block that contains FMRecord

        // Create NFRecord object from packed NFRecord
        NFRecordCreateFromMemory(nfRecord, nfRecordSize, 0, NULL, &hNFRecord);
        // Create FMRecord object from NFRecord object
        FMRecordCreateFromNFRecord(hNFRecord, 0, &hFMRecord);
        // Delete NFRecord object
        NFRecordFree(hNFRecord);
        // Calculate size of FMRecord object to store in memory
        FMRecordGetSize(hFMRecord, 0, &fmRecordSize);
        // - or -
        //FMRecordGetSize(hFMRecord, FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Allocate memory for FMRecord
        fmRecord = NAlloc(fmRecordSize);
        // Store FMRecord object in memory
        FMRecordSaveToMemory(hFMRecord, fmRecord, fmRecordSize, 0, &fmRecordSize);
        // - or -
        //FMRecordSaveToMemory(hFMRecord, fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // ...
        // Do something with FMRecord
        // ...
        // Free memory allocated for FMRecord
        NFree(fmRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
```

```
        public void PackedNFRecordToStoredFMRecord(
                byte[] packedNFRecord // packed NFRecord
        )
        {
                // Create NFRecord object from packed NFRecord
                NFRecord nfRecord = new NFRecord(packedNFRecord);
                // Create FMRecord object from NFRecord object
                FMRecord fmRecord = new FMRecord(nfRecord);
                // Store FMRecord object in memory
                byte[] storedFMRecord = fmRecord.Save();
                // - or -
                //byte[] storedFMRecord = fmRecord.Save(
                // FmrFingerView.FlagUseNeurotecFields);
                // ...
                // Do something with FMRecord
                // ...
        }
}
```

Remarks

The created FMRecord has Product Owner part of CBEFF Product ID set to Neurotechnologija CBEFF Format Owner value (`0x0031`). Also additional fields of Neurotechnologija are created for FMRecord Finger View and filled with values from given NFRecord to ensure minimal loss of information if the FMRecord will be later converted back to NFRecord. If it will not be converted back to NFRecord and/or minimal FMRecord size is desired then `FMRFV_SKIP_NEUROTEC_FIELDS` (FmrFingerView .`FlagSkipNeurotecFields` in .NET) flag should be used.

If CBEFF Product ID is changed then additional Neurotechnologija fields are not created by default. To force their creation use commented lines(s).

3.2.2. Converting Version 1.0 Format NFRecord to FMRecord

Converting version 1.0 format NFRecord to FMRecord is identical to converting NFRecord to FMRecord. See Section 3.2.1, “Converting NFRecord to FMRecord”.

3.2.3. Converting Two Version 1.0 Format NFRecords to FMRecord

```
[C]

#include "FMRecord.h"

void PackedNFRecordV1ToStoredFMRecordTwoFinger(
        NSizeType nfRecordSize1, // size of first packed NFRecord
        const void * nfRecord1, // pointer to memory block
                // that contains first packed NFRecord
        NSizeType nfRecordSize2, // size of second packed NFRecord
```

```
        const void * nfRecord2 // pointer to memory block
                // that contains second packed NFRecord
)
{
        HNFTemplate hNFTemplate; // handle to NFTemplate object
        HNFRecord hNFRecord1; // handle to first NFRecord object
        HNFRecord hNFRecord2; // handle to second NFRecord object
        HFMRecord hFMRecord; // handle to FMRecord object
        NSizeType fmRecordSize; // size of FMRecord
        void * fmRecord; // pointer to memory block that contains FMRecord

        // Create NFTemplate object
        NFTemplateCreate(&hNFTemplate);
        // Add records created from first and second packed NFRecords
        // to NFTemplate object
        NFTemplateAddRecordFromMemory(hNFTemplate, nfRecord1, nfRecordSize1,
                0, &hNFRecord1);
        NFTemplateAddRecordFromMemory(hNFTemplate, nfRecord2, nfRecordSize2,
                0, &hNFRecord2);
        // Create FMRecord object from NFTemplate object
        FMRecordCreateFromNFTemplate(hNFTemplate, 0, &hFMRecord);
        // Delete NFTemplate object
        NFTemplateFree(hNFTemplate);
        // Calculate size of FMRecord object to store in memory
        FMRecordGetSize(hFMRecord, 0, &fmRecordSize);
        // - or -
        //FMRecordGetSize(hFMRecord, FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Allocate memory for FMRecord
        fmRecord = NAlloc(fmRecordSize);
        // Store FMRecord object in memory
        FMRecordSaveToMemory(hFMRecord, fmRecord, fmRecordSize, 0, &fmRecordSize);
        // - or -
        //FMRecordSaveToMemory(hFMRecord, fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // ...
        // Do something with FMRecord
        // ...
        // Free memory allocated for FMRecord
        NFree(fmRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void PackedNFRecordToStoredFMRecord(
                byte[] packedNFRecord1, // first packed NFRecord
                byte[] packedNFRecord2 // second packed NFRecord
        )
        {
```

```
                // Create NFTemplate object
                NFTemplate nfTemplate = new NFTemplate();
                // Add record created from first and second packed NFRecords
                // to NFTemplate object
                NFRecord nfRecord1 = nfTemplate.Records.Add(packedNFRecord1);
                NFRecord nfRecord2 = nfTemplate.Records.Add(packedNFRecord2);
                // Create FMRecord object from NFTemplate object
                FMRecord fmRecord = new FMRecord(nfTemplate);
                // Store FMRecord object in memory
                byte[] storedFMRecord = fmRecord.Save();
                // - or -
                //byte[] storedFMRecord = fmRecord.Save(
                // FmrFingerView.FlagUseNeurotecFields);
                // ...
                // Do something with FMRecord
                // ...
        }
}
```

Remarks

The created FMRecord has Product Owner part of CBEFF Product ID set to Neurotechnologija CBEFF Format Owner value (`0x0031`). Also additional fields of Neurotechnologija are created for FMRecord Finger View and filled with values from given NFRecord to ensure minimal loss of information if the FMRecord will be later converted back to NFRecord. If it will not be converted back to NFRecord and/or minimal FMRecord size is desired then `FMRFV_SKIP_NEUROTEC_FIELDS` (FmrFingerView .`FlagSkipNeurotecFields` in .NET) flag should be used.

If CBEFF Product ID is changed then additional Neurotechnologija fields are not created by default. To force their creation use commented lines(s).

3.2.4. Converting NTemplate to FMRecord

```
[C]

#include "FMRecord.h"

void PackedNTemplateToStoredFMRecord(
        NSizeType nTemplateSize, // size of packed NTemplate
        const void * nTemplate // pointer to memory block
                // that contains packed NTemplate
)
{
        HNTemplate hNTemplate; // handle to NTemplate object
        HNFTemplate hNFTemplate; // handle to NFTemplate object
        HFMRecord hFMRecord; // handle to FMRecord object
        NSizeType fmRecordSize; // size of FMRecord
        void * fmRecord; // pointer to memory block that contains FMRecord

        // Create NTemplate object from packed NTemplate
        NTemplateCreateFromMemory(nTemplate, nTemplateSize, 0, NULL, &hNTemplate);
```

```
        // Retrieve NFTemplate object from NTemplate object
        NTemplateGetFingers(hNTemplate, &hNFTemplate);
        // Create FMRecord object from NFTemplate object
        FMRecordCreateFromNFTemplate(hNFTemplate, 0, &hFMRecord);
        // Delete NTemplate object
        NTemplateFree(hNTemplate);
        // Calculate size of FMRecord object to store in memory
        FMRecordGetSize(hFMRecord, 0, &fmRecordSize);
        // - or -
        //FMRecordGetSize(hFMRecord, FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Allocate memory for FMRecord
        fmRecord = NAlloc(fmRecordSize);
        // Store FMRecord object in memory
        FMRecordSaveToMemory(hFMRecord, fmRecord, fmRecordSize, 0, &fmRecordSize);
        // - or -
        //FMRecordSaveToMemory(hFMRecord, fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // ...
        // Do something with FMRecord
        // ...
        // Free memory allocated for FMRecord
        NFree(fmRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void PackedNTemplateToStoredFMRecord(
                byte[] packedNTemplate // packed NTemplate
        )
        {
                // Create NTemplate object from packed NTemplate
                NTemplate nTemplate = new NTemplate(packedNTemplate);
                // Retrieve NFTemplate object from NTemplate object
                NFTemplate nfTemplate = nTemplate.Fingers;
                // Create FMRecord object from NFTemplate object
                FMRecord fmRecord = new FMRecord(nfTemplate);
                // Store FMRecord object in memory
                byte[] storedFMRecord = fmRecord.Save();
                // - or -
                //byte[] storedFMRecord = fmRecord.Save(
                // FmrFingerView.FlagUseNeurotecFields);
                // ...
                // Do something with FMRecord
                // ...
        }
}
```

Remarks

The created FMRecord has Product Owner part of CBEFF Product ID set to Neurotechnologija CBEFF Format Owner value (`0x0031`). Also additional fields of Neurotechnologija are created for each FMRecord Finger View and filled with values from appropriate NFRecord stored in NFTemplate of NTemplate to ensure minimal loss of information if the FMRecord will be later converted back to NTemplate. If it will not be converted back to NTemplate and/or minimal FMRecord size is desired then `FMRFV_SKIP_NEUROTEC_FIELDS` (FmrFingerView `.FlagSkipNeurotecFields` in .NET) flag should be used.

If CBEFF Product ID is changed then additional Neurotechnologija fields are not created by default. To force their creation use commented lines(s).

3.2.5. Converting FMRecord to NFRecord

```
[C]

#include "FMRecord.h"

void StoredFMRecordToPackedNFRecord(
        NSizeType fmRecordSize, // size of FMRecord
        const void * fmRecord // pointer to memory block that contains FMRecord
)
{
        HFMRecord hFMRecord; // handle to FMRecord object
        HFmrFingerView hFmrFingerView; // handle to FmrFingerView object
        HNFRecord hNFRecord; // handle to NFRecord object
        NSizeType nfRecordSize; // size of packed NFRecord
        void * nfRecord; // pointer to memory block
                // that contains packed NFRecord

        // Create FMRecord object from FMRecord stored in memory
        FMRecordCreateFromMemory(fmRecord, fmRecordSize, 0, NULL, &hFMRecord);
        // - or -
        //FMRecordCreateFromMemory(fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, NULL, &hFMRecord);
        // Retrieve first FmrFingerView object from FMRecord object
        FMRecordGetFingerView(hFMRecord, 0, &hFmrFingerView);
        // Convert FmrFingerView object to NFRecord object
        FmrFingerViewToNFRecord(hFmrFingerView, 0, &hNFRecord);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // Calculate packed size of NFRecord object
        NFRecordGetSize(hNFRecord, 0, &nfRecordSize);
        // Use the following line instead of previous one
        // if it is needed to pack NFRecord in version 1.0 format
        //NFRecordGetSizeV1(hNFRecord, 0, &nfRecordSize);
        // Allocate memory for packed NFRecord
        nfRecord = NAlloc(nfRecordSize);
        // Pack NFRecord object
        NFRecordSaveToMemory(hNFRecord, nfRecord, nfRecordSize, 0, &nfRecordSize);
        // Use the following line instead of previous one
        // if it is needed to pack NFRecord in version 1.0 format
```

```
        //NFRecordSaveToMemoryV1(hNFRecord, nfRecord, nfRecordSize, 0,
        // &nfRecordSize);
        // Delete NFRecord object
        NFRecordFree(hNFRecord);
        // ...
        // Do something with packed NFRecord
        // ...
        // Free memory allocated for packed NFRecord
        NFree(nfRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void StoredFMRecordToPackedNFRecord(
                byte[] storedFMRecord // FMRecord stored in memory
        )
        {
                // Create FMRecord object from FMRecord stored in memory
                FMRecord fmRecord = new FMRecord(storedFMRecord);
                // - or -
                //FMRecord fmRecord = new FMRecord(storedFMRecord,
                // FmrFingerView.UseNeurotecFields);
                // Retrieve first FmrFingerView object from FMRecord object
                FmrFingerView fmrFingerView = fmRecord.FingerViews[0];
                // Convert FmrFingerView object to NFRecord object
                NFRecord nfRecord = fmrFingerView.ToNFRecord();
                // Pack NFRecord object
                byte[] packedNFRecord = nfRecord.Save();
                // Use the following line instead of previous one
                // if it is needed to pack NFRecord in version 1.0 format
                //byte[] packedNFRecord = nfRecord.SaveV1();
                // ...
                // Do something with packed NFRecord
                // ...
        }
}
```

Remarks

Commented line has to be used if FMRecord was converted from NFRecord and Product Owner part of CBEFF Product ID is set to value different from Neurotechnologija CBEFF Format Owner value and minimal information loss is desired. See also Section 3.2.1, “Converting NFRecord to FMRecord”.

3.2.6. Converting Two-Finger FMRecord to NFRecords in Version 1.0 Format

[C]

```
#include "FMRecord.h"

void StoredFMRecordToPackedNFRecordV1TwoFinger(
        NSizeType fmRecordSize, // size of FMRecord
        const void * fmRecord // pointer to memory block that contains FMRecord
)
{
        HFMRecord hFMRecord; // handle to FMRecord object
        HNFTemplate hNFTemplate; // handle to NFTemplate object
        HNFRecord hNFRecord1; // handle to first NFRecord object
        HNFRecord hNFRecord2; // handle to second NFRecord object
        NSizeType nfRecordSize1; // size of first packed NFRecord
        NSizeType nfRecordSize2; // size of second packed NFRecord
        void * nfRecord1; // pointer to memory block
                // that contains first packed NFRecord
        void * nfRecord2; // pointer to memory block
                // that contains second packed NFRecord

        // Create FMRecord object from FMRecord stored in memory
        FMRecordCreateFromMemory(fmRecord, fmRecordSize, 0, NULL, &hFMRecord);
        // - or -
        //FMRecordCreateFromMemory(fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, NULL, &hFMRecord);
        // Convert FMRecord object to NFTemplate object
        FMRecordToNFTemplate(hFMRecord, 0, &hNFTemplate);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // Retreive first and second NFRecord objects from NFTemplate
        NFTemplateGetRecord(hNFTemplate, 0, &hNFRecord1);
        NFTemplateGetRecord(hNFTemplate, 1, &hNFRecord2);
        // Calculate packed sizes of first and second NFRecord objects
        // in v 1.0 format
        NFRecordGetSizeV1(hNFRecord1, 0, &nfRecordSize1);
        NFRecordGetSizeV1(hNFRecord2, 0, &nfRecordSize2);
        // Allocate memory for first and second packed NFRecords
        nfRecord1 = NAlloc(nfRecordSize1);
        nfRecord2 = NAlloc(nfRecordSize2);
        // Pack first and second NFRecord objects in v 1.0 format
        NFRecordSaveToMemoryV1(hNFRecord1, nfRecord1, nfRecordSize1,
                0, &nfRecordSize1);
        NFRecordSaveToMemoryV1(hNFRecord2, nfRecord2, nfRecordSize2,
                0, &nfRecordSize2);
        // Delete NFTemplate object
        NFTemplateFree(hNFTemplate);
        // ...
        // Do something with packed NFRecords
        // ...
        // Free memory allocated for first and second packed NFRecords
        NFree(nfRecord1);
        NFree(nfRecord2);
}
```

[C#]

```
using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void StoredFMRecordToPackedNFRecord(
                byte[] storedFMRecord // FMRecord stored in memory
        )
        {
                // Create FMRecord object from FMRecord stored in memory
                FMRecord fmRecord = new FMRecord(storedFMRecord);
                // - or -
                //FMRecord fmRecord = new FMRecord(storedFMRecord,
                // FmrFingerView.UseNeurotecFields);
                // Convert FMRecord object to NFTemplate object
                NFTemplate nfTemplate = fmRecord.ToNFTemplate();
                // Retreive first and second NFRecord objects
                // from NFTemplate object
                NFRecord nfRecord1 = nfTemplate.Records[0];
                NFRecord nfRecord2 = nfTemplate.Records[1];
                // Pack first and second NFRecord objects in version 1.0 format
                byte[] packedNFRecord1 = nfRecord1.SaveV1();
                byte[] packedNFRecord2 = nfRecord2.SaveV1();
                // ...
                // Do something with packed NFRecords
                // ...
        }
}
```

Remarks

Commented line has to be used if FMRecord was converted from NFRecord and Product Owner part of CBEFF Product ID is set to value different from Neurotechnologija CBEFF Format Owner value and minimal information loss is desired. See also Section 3.2.1, “Converting NFRecord to FMRecord”.

3.2.7. Converting FMRecord to NTemplate

```
[C]

#include "FMRecord.h"

void StoredFMRecordToPackedNTemplate(
        NSizeType fmRecordSize, // size of FMRecord
        const void * fmRecord // pointer to memory block that contains FMRecord
)
{
        HFMRecord hFMRecord; // handle to FMRecord object
        HNTemplate hNTemplate; // handle to NTemplate object
        NSizeType nTemplateSize; // size of packed NTemplate
        void * nTemplate; // pointer to memory block
                // that contains packed NTemplate
```

```
        // Create FMRecord object from FMRecord stored in memory
        FMRecordCreateFromMemory(fmRecord, fmRecordSize, 0, NULL, &hFMRecord);
        // - or -
        //FMRecordCreateFromMemory(fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, NULL, &hFMRecord);
        // Convert FMRecord object to NTemplate object
        FMRecordToNTemplate(hFMRecord, 0, &hNTemplate);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // Calculate packed size of NTemplate object
        NTemplateGetSize(hNTemplate, 0, &nTemplateSize);
        // Allocate memory for packed NTemplate
        nTemplate = NAlloc(nTemplateSize);
        // Pack NTemplate object
        NTemplateSaveToMemory(hNTemplate, nTemplate, nTemplateSize, 0,
                &nTemplateSize);
        // Delete NTemplate object
        NTemplateFree(hNTemplate);
        // ...
        // Do something with packed NTemplate
        // ...
        // Free memory allocated for packed NTemplate
        NFree(nTemplate);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void StoredFMRecordToPackedNTemplate(
                byte[] storedFMRecord // FMRecord stored in memory
        )
        {
                // Create FMRecord object from FMRecord stored in memory
                FMRecord fmRecord = new FMRecord(storedFMRecord);
                // - or -
                //FMRecord fmRecord = new FMRecord(storedFMRecord,
                // FmrFingerView.UseNeurotecFields);
                // Convert FMRecord object to NTemplate object
                NTemplate nTemplate = fmRecord.ToNTemplate();
                // Pack NTemplate object
                byte[] packedNTemplate = nTemplate.Save();
                // ...
                // Do something with packed NTemplate
                // ...
        }
}
```

Remarks

Commented line has to be used if FMRecord was converted from NTemplate and Product Owner part of CBEFF Product ID is set to value different from Neurotechnologija CBEFF Format Owner value and minimal information loss is desired. See also Section 3.2.4, “Converting NTemplate to FMRecord”.

3.2.8. Converting NFRecord to ANTemplate

```
[C]

#include "ANTemplate.h"

void PackedNFRecordToANTemplateFile(
        NSizeType nfRecordSize, // size of packed NFRecord
        const void * nfRecord, // pointer to memory block
                // that contains packed NFRecord
        const NAChar * szTot, // pointer to string that specifies
                // type of transaction
        const NAChar * szDai, // pointer to string that specifies
                // destination agency identifier
        const NAChar * szOri, // pointer to string that specifies
                // originating agency identifier
        const NAChar * szTcn, // pointer to string that specifies
                // transaction control number
        const NChar * szFileName // pointer to string that specifies
                // filename to store ANTemplate
)
{
        HNFRecord hNFRecord; // handle to NFRecord object
        HANTemplate hANTemplate; // handle to ANTemplate object

        // Create NFRecord object from packed NFRecord
        NFRecordCreateFromMemory(nfRecord, nfRecordSize, 0, NULL, &hNFRecord);
        // Create ANTemplate object from NFRecord object
        ANTemplateCreateFromNFRecord(hNFRecord,
                szTot, szDai, szOri, szTcn, &hANTemplate);
        // Delete NFRecord object
        NFRecordFree(hNFRecord);
        // Store ANTemplate object in file
        ANTemplateSaveToFile(hANTemplate, szFileName);
        // Delete ANTemplate object
        ANTemplateFree(hANTemplate);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void PackedNFRecordToANTemplateFile(
                byte[] packedNFRecord, // packed NFRecord
                string tot, // type of transaction
                string dai, // destination agency identifier
```

```
                string ori, // originating agency identifier
                string tcn, // transaction control number
                string fileName // filename to store ANTemplate
        )
        {
                // Create NFRecord object from packed NFRecord
                NFRecord nfRecord = new NFRecord(packedNFRecord);
                // Create ANTemplate object from NFRecord object
                ANTemplate anTemplate = new ANTemplate(
                        nfRecord, tot, dai, ori, tcn);
                // Store ANTemplate object in file
                anTemplate.Save(fileName);
        }
}
```

3.2.9. Converting Version 1.0 Format NFRecord to ANTemplate

Converting version 1.0 format NFRecord to ANTemplate is identical to converting NFRecord to ANTemplate. See Section 3.2.8, “Converting NFRecord to ANTemplate”.

3.2.10. Converting NTemplate to ANTemplate

```
[C]

#include "ANTemplate.h"

void PackedNTemplateToANTemplateFile(
        NSizeType nTemplateSize, // size of packed NTemplate
        const void * nTemplate, // pointer to memory block
                // that contains packed NTemplate
        const NAChar * szTot, // pointer to string that specifies
                // type of transaction
        const NAChar * szDai, // pointer to string that specifies
                // destination agency identifier
        const NAChar * szOri, // pointer to string that specifies
                // originating agency identifier
        const NAChar * szTcn, // pointer to string that specifies
                // transaction control number
        const NChar * szFileName // pointer to string that specifies
                // filename to store ANTemplate
)
{
        HNTemplate hNTemplate; // handle to NTemplate object
        HANTemplate hANTemplate; // handle to ANTemplate object

        // Create NTemplate object from packed NTemplate
        NTemplateCreateFromMemory(nTemplate, nTemplateSize, 0, NULL, &hNTemplate);
        // Create ANTemplate object from NFRecord object
        ANTemplateCreateFromNTemplate(hNTemplate,
                szTot, szDai, szOri, szTcn, &hANTemplate);
        // Delete NTemplate object
        NTemplateFree(hNTemplate);
```

```
        // Store ANTemplate object in file
        ANTemplateSaveToFile(hANTemplate, szFileName);
        // Delete ANTemplate object
        ANTemplateFree(hANTemplate);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void PackedNTemplateToANTemplateFile(
                byte[] packedNTemplate, // packed NTemplate
                string tot, // type of transaction
                string dai, // destination agency identifier
                string ori, // originating agency identifier
                string tcn, // transaction control number
                string fileName // filename to store ANTemplate
        )
        {
                // Create NTemplate object from packed NTemplate
                NTemplate nTemplate = new NTemplate(packedNTemplate);
                // Create ANTemplate object from NTemplate object
                ANTemplate anTemplate = new ANTemplate(
                        nTemplate, tot, dai, ori, tcn);
                // Store ANTemplate object in file
                anTemplate.Save(fileName);
        }
}
```

3.2.11. Converting ANTemplate to NFRecord

```
[C]

#include "ANTemplate.h"

void ANTemplateFileToPackedNFRecord(
        const NChar * szFileName // pointer to string that specifies
                // name of file were ANTemplate is stored
)
{
        HANTemplate hANTemplate; // handle to ANTemplate object
        HNFTemplate hNFTemplate; // handle to NFTemplate object
        HNFRecord hNFRecord; // handle to NFRecord object
        NSizeType nfRecordSize; // size of packed NFRecord
        void * nfRecord; // pointer to memory block
                // that contains packed NFRecord

        // Create ANTemplate object from file
        ANTemplateCreateFromFile(szFileName, &hANTemplate);
        // Convert ANTemplate object to NTemplate object
```

```
        ANTemplateToNFTemplate(hANTemplate, &hNFTemplate);
        // Delete ANTemplate object
        ANTemplateFree(hANTemplate);
        // Retrieve first NFRecord object from NFTemplate object
        NFTemplateGetRecord(hNFTemplate, 0, &hNFRecord);
        // Calculate packed size of NFRecord object
        NFRecordGetSize(hNFRecord, 0, &nfRecordSize);
        // Use the following line instead of previous one
        // if it is needed to pack NFRecord in version 1.0 format
        //NFRecordGetSizeV1(hNFRecord, 0, &nfRecordSize);
        // Allocate memory for packed NFRecord
        nfRecord = NAlloc(nfRecordSize);
        // Pack NFRecord object
        NFRecordSaveToMemory(hNFRecord, nfRecord, nfRecordSize, 0, &nfRecordSize);
        // Use the following line instead of previous one
        // if it is needed to pack NFRecord in version 1.0 format
        //NFRecordSaveToMemoryV1(hNFRecord, nfRecord, nfRecordSize, 0,
        // &nfRecordSize);
        // Delete NFTemplate object
        NFTemplateFree(hNFTemplate);
        // ...
        // Do something with packed NFRecord
        // ...
        // Free memory allocated for packed NFRecord
        NFree(nfRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void ANTemplateFileToPackedNFRecord(
                string fileName // name of file were ANTemplate is stored
        )
        {
                // Create ANTemplate object from file
                ANTemplate anTemplate = new ANTemplate(fileName);
                // Convert ANTemplate object to NFTemplate object
                NFTemplate nfTemplate = anTemplate.ToNFTemplate();
                // Retrieve first NFRecord object from NFTemplate object
                NFRecord nfRecord = nfTemplate.Records[0];
                // Pack NFRecord object
                byte[] packedNFRecord = nfRecord.Save();
                // Use the following line instead of previous one
                // if it is needed to pack NFRecord in version 1.0 format
                //byte[] packedNFRecord = nfRecord.SaveV1();
                // ...
                // Do something with packed NFRecord
                // ...
        }
}
```

3.2.12. Converting ANTemplate to NTemplate

```
[C]

#include "ANTemplate.h"

void ANTemplateFileToPackedNTemplate(
        const NChar * szFileName // pointer to string that specifies
                // name of file were ANTemplate is stored
)
{
        HANTemplate hANTemplate; // handle to ANTemplate object
        HNTemplate hNTemplate; // handle to NTemplate object
        NSizeType nTemplateSize; // size of packed NTemplate
        void * nTemplate; // pointer to memory block
                // that contains packed NTemplate

        // Create ANTemplate object from file
        ANTemplateCreateFromFile(szFileName, &hANTemplate);
        // Convert ANTemplate object to NTemplate object
        ANTemplateToNTemplate(hANTemplate, &hNTemplate);
        // Delete ANTemplate object
        ANTemplateFree(hANTemplate);
        // Calculate packed size of NTemplate object
        NTemplateGetSize(hNTemplate, 0, &nTemplateSize);
        // Allocate memory for packed NTemplate
        nTemplate = NAlloc(nTemplateSize);
        // Pack NTemplate object
        NTemplateSaveToMemory(hNTemplate, nTemplate, nTemplateSize, 0,
                &nTemplateSize);
        // Delete NTemplate object
        NTemplateFree(hNTemplate);
        // ...
        // Do something with packed NTemplate
        // ...
        // Free memory allocated for packed NTemplate
        NFree(nTemplate);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void ANTemplateFileToPackedNTemplate(
                string fileName // name of file were ANTemplate is stored
        )
        {
                // Create ANTemplate object from file
                ANTemplate anTemplate = new ANTemplate(fileName);
                // Convert ANTemplate object to NTemplate object
                NTemplate nTemplate = anTemplate.ToNTemplate();
                // Pack NTemplate object
```

```
                byte[] packedNTemplate = nTemplate.Save();
                // ...
                // Do something with packed NTemplate
                // ...
        }
}
```

Chapter 4. Reference (C/C++)

This chapter contains reference of all libraries included in Template Management and Conversion Add-On for developers using C/C++ language.

Libraries

ANTemplate	Provides functionality for converting ANSI/NIST-ITL 1-2000 standard files to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.
FMRecord	Provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.
NCore	Provides infrastructure for Neurotechnologija components.
NFRecord	Provides functionality for packing, unpacking and editing Neurotechnologija Finger Records.
NFTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates.
NTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

4.1. ANTemplate Library

Provides functionality for converting ANSI/NIST-ITL 1-2000 standard files to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

Import library (Windows): `ANTemplate.dll.lib`.

DLL (Windows): `ANTemplate.dll`.

Shared object (Linux): `libANTemplate.so`.

Requirements (Windows):

- `NCore.dll`.
- `NFRecord.dll`.
- `NFTemplate.dll`.
- `NTemplate.dll`.

Requirements (Linux):

- `libNCore.so`.
- `libNFRecord.so`.
- `libNFTemplate.so`.
- `libNTemplate.so`.

Modules

ANTemplate	Provides functionality for converting ANSI/NIST-ITL 1-2000 standard (ANTemplate) files to and/or from Neurotechnologija Finger Records (NFRecords), Neurotechnologija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).

4.1.1. ANTemplate Module

Provides functionality for converting ANSI/NIST-ITL 1-2000 standard (ANTemplate) files to and/or from Neurotechnologija Finger Records (NFRecords), Neurotechnologija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).

Header file: `ANTemplate.h`.

Functions

`ANTemplateCreateFromFile`	Loads an ANTemplate from the specified file.
`ANTemplateCreateFromNFRecord`	Creates an ANTemplate from the specified NFRecord.
`ANTemplateCreateFromNFTemplate`	Creates an ANTemplate from the specified NFTemplate.
`ANTemplateCreateFromNTemplate`	Creates an ANTemplate from the specified NTemplate.
`ANTemplateFree`	Deletes the ANTemplate. After the object is

	deleted the specified handle is no longer valid.
ANTemplateSaveToFile	Saves the ANTemplate to the specified file.
ANTemplateToNFTemplate	Converts the ANTemplate to a NFTemplate.
ANTemplateToNTemplate	Converts the ANTemplate to a NTemplate.

Types

HANTemplate	Handle to an ANTemplate object.

Macros

ANT_MAX_DIMENSION	For internal use.

See Also

ANTemplate Library

4.1.1.1. ANTemplateCreateFromFile Function

Loads an ANTemplate from the specified file.

```
NResult N_API ANTemplateCreateFromFile(
        const NChar * szFileName,
        HANTemplate * pHANTemplate
);
```

Parameters

szFileName	[in] Pointer to a string that specifies file-name.
pHANTemplate	[in] Pointer to a HANTemplate that receives handle to newly created ANTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHANTemplate* or *szFileName* is NULL.
N_E_IO	Input/output error has occurred.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `ANTemplateFree` function.

See Also

ANTemplate Module | HANTemplate | `ANTemplateFree` | `ANTemplateSaveToFile`

4.1.1.2. ANTemplateCreateFromNFRecord Function

Creates an ANTemplate from the specified NFRecord.

```
NResult N_API ANTemplateCreateFromNFRecord(
        HNFRecord hNFRecord,
        const NChar * szTot,
        const NChar * szDai,
        const NChar * szOri,
        const NChar * szTcn,
        HANTemplate * pHANTemplate
);
```

Parameters

hNFRecord	[in] Handle to the NFRecord object.
szTot	[in] Pointer to a string that specifies Type Of Transaction
szDai	[in] Pointer to a string that specifies Destination Agency Identifier
szOri	[in] Pointer to a string that specifies Originating Agency Identifier
szTcn	[in] Pointer to a string that specifies Transaction Control Number
pHANTemplate	[in] Pointer to a HANTemplate that receives

	handle to newly created ANTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hNFRecord*, *szTot*, *szDai*, *szOri*, *szTcn* or *pHANTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `ANTemplateFree` function.

See Also

ANTemplate Module | HANTemplate | `ANTemplateFree` | HNFRecord

4.1.1.3. ANTemplateCreateFromNFTemplate Function

Creates an ANTemplate from the specified NFTemplate.

```
NResult N_API ANTemplateCreateFromNFTemplate(
        HNFTemplate hNFTemplate,
        const NChar * szTot,
        const NChar * szDai,
        const NChar * szOri,
        const NChar * szTcn,
        HANTemplate * pHANTemplate
);
```

Parameters

hNFTemplate	[in] Handle to the NFTemplate object.
szTot	[in] Pointer to a string that specifies Type Of Transaction
szDai	[in] Pointer to a string that specifies Destination Agency Identifier
szOri	[in] Pointer to a string that specifies Origin-

	ating Agency Identifier
szTcn	[in] Pointer to a string that specifies Transaction Control Number
pHANTemplate	[in] Pointer to a HANTemplate that receives handle to newly created ANTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hNFTemplate*, *szTot*, *szDai*, *szOri*, *szTcn* or *pHANTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `ANTemplateFree` function.

See Also

ANTemplate Module | HANTemplate | `ANTemplateFree` | HNFTemplate | `ANTemplateToNFTemplate`

4.1.1.4. ANTemplateCreateFromNTemplate Function

Creates an ANTemplate from the specified NTemplate.

```
NResult N_API ANTemplateCreateFromNTemplate(
        HNTemplate hNTemplate,
        const NChar * szTot,
        const NChar * szDai,
        const NChar * szOri,
        const NChar * szTcn,
        HANTemplate * pHANTemplate
);
```

Parameters

hNTemplate	[in] Handle to the NTemplate object.

szTot	[in] Pointer to a string that specifies Type Of Transaction
szDai	[in] Pointer to a string that specifies Destination Agency Identifier
szOri	[in] Pointer to a string that specifies Originating Agency Identifier
szTcn	[in] Pointer to a string that specifies Transaction Control Number
pHANTemplate	[in] Pointer to a HANTemplate that receives handle to newly created ANTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hNTemplate*, *szTot*, *szDai*, *szOri*, *szTcn* or *pHANTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `ANTemplateFree` function.

See Also

ANTemplate Module | HANTemplate | `ANTemplateFree` | HNTemplate | `ANTemplateToNTemplate`

4.1.1.5. ANTemplateFree Function

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

```
void N_API ANTemplateFree(
        HANTemplate hANTemplate
);
```

Parameters

hANTemplate	[in] Handle to the ANTemplate object.

Remarks

If *hANTemplate* is NULL, does nothing.

See Also

ANTemplate Module | HANTemplate

4.1.1.6. ANTemplateSaveToFile Function

Saves the ANTemplate to the specified file.

```
NResult N_API ANTemplateSaveToFile(
        HANTemplate hANTemplate,
        const NChar * szFileName
);
```

Parameters

hANTemplate	[in] Handle to the ANTemplate object.
szFileName	[in] Pointer to a 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	*hANTemplate* or *szFileName* is NULL.
N_E_IO	Input/output error has occurred.

See Also

ANTemplate Module | HANTemplate | ANTemplateCreateFromFile

4.1.1.7. ANTemplateToNFTemplate Function

Copyright © 2006 Neurotechnologija

Converts the ANTemplate to a NFTemplate.

```
NResult N_API ANTemplateToNFTemplate(
        HANTemplate hANTemplate,
        HNFTemplate * pHNFTemplate
);
```

Parameters

hANTemplate	[in] Handle to the ANTemplate object.
pHNFTemplate	[out] Pointer to a HNFTemplate that receives handle to newly created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hANTemplate* or *pHNFTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

ANTemplate Module | HANTemplate | HNFTemplate | ANTemplateCreateFromNFTemplate

4.1.1.8. ANTemplateToNTemplate Function

Converts the ANTemplate to a NTemplate.

```
NResult N_API ANTemplateToNTemplate(
        HANTemplate hANTemplate,
        HNTemplate * pHNTemplate
);
```

Parameters

hANTemplate	[in] Handle to the ANTemplate object.

pHNTemplate	[out] Pointer to a HNTemplate that receives handle to newly created NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hANTemplate* or *pHNTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

ANTemplate Module | HANTemplate | HNTemplate | ANTemplateCreateFromNTemplate

4.2. FMRecord Library

Provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

Import library (Windows): FMRecord.dll.lib.

DLL (Windows): FMRecord.dll.

Shared object (Linux): libFMRecord.so.

Requirements (Windows):

- NCore.dll.
- NFRecord.dll.
- NFTemplate.dll.
- NTemplate.dll.

Requirements (Linux):

- libNCore.so.
- libNFRecord.so.
- libNFTemplate.so.

- `libNTemplate.so`.

Modules

FMRecord	Provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates (FMRecords) to and/or from Neurotechnologija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).
FmrFingerView	Provides functionality for editing and converting ANSI INCITS 378-2004 standard templates finger views (FmrFingerViews) to Neurotechnologija Finger Records (NFRecords).

4.2.1. FMRecord Module

Provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates (FMRecords) to and/or from Neurotechnologija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).

Header file: `FMRecord.h`.

Functions

`FMRecordAddFingerView`	Adds an empty finger view to the end of FMRecord finger views.
`FMRecordAddFingerViewCopy`	Adds a copy of the specified finger view to the end of FMRecord finger views.
`FMRecordCalculateSize`	Calculates the size of a stored FMRecord containing the specified number of finger views of specified size.
`FMRecordClearFingerViews`	Removes all finger views from the FMRecord.
`FMRecordClone`	Creates a copy of the FMRecord.
`FMRecordCreate`	Creates an empty FMRecord.
`FMRecordCreateFromMemory`	Restores a FMRecord from the specified memory buffer.
`FMRecordCreateFromNFRecord`	Creates a FMRecord from the specified

	NFRecord.
FMRecordCreateFromNFTemplate	Creates a FMRecord from the specified NFTemplate.
FMRecordFree	Deletes the FMRecord. After the object is deleted the specified handle is no longer valid.
FMRecordGetCaptureEquipmentCompliance	Retrieves the Capture Equipment Compliance of the FMRecord.
FMRecordGetCaptureEquipmentId	Retrieves the Capture Equipment Id from of the FMRecord.
FMRecordGetCbeffProductId	Retrieves the CBEFF Product Id of the FMRecord.
FMRecordGetFingerView	Retrieves the finger view at the specified index of the FMRecord.
FMRecordGetFingerViewCapacity	Retrieves the number of finger views that the FMRecord can contain.
FMRecordGetFingerViewCount	Retrieves the number of finger views in the FMRecord.
FMRecordGetResolutionX	Retrieves horizontal resolution of the image the FMRecord is made from.
FMRecordGetResolutionY	Retrieves vertical resolution of the image the FMRecord is made from.
FMRecordGetSize	Calculates stored size of the FMRecord.
FMRecordGetSizeX	Retrieves horizontal size of the image the FMRecord is made from.
FMRecordGetSizeY	Retrieves vertical size of the image the FMRecord is made from.
FMRecordInfoDispose	For internal use.
FMRecordRemoveFingerView	Removes the finger view at the specified index of the FMRecord.
FMRecordSaveToMemory	Stores the FMRecord in the specified memory buffer.
FMRecordSetCaptureEquipmentCompliance	Sets the Capture Equipment Compliance of the FMRecord.
FMRecordSetCaptureEquipmentId	Sets the Capture Equipment Id of the FM-

Copyright © 2006 Neurotechnologija

	Record.
FMRecordSetCbeffProductId	Sets the CBEFF Product Id of the FMRecord.
FMRecordSetFingerViewCapacity	Sets the number of finger views that the FMRecord can contain.
FMRecordToNFTemplate	Converts the FMRecord to a NFTemplate.
FMRecordToNTemplate	Converts the FMRecord to a NTemplate.

Structures

FMRecordInfo	For internal use.

Types

HFMRecord	Handle to FMRecord object.

Macros

FMR_MAX_FINGER_VIEW_COUNT	The maximum number of finger views FMRecord can contain.
FMR_PROCESS_FIRST_FINGER_VIEW_ONLY	The flag indicating whether only the first finger view should be loaded or saved while loading or saving FMRecord.

See Also

FMRecord Library

4.2.1.1. FMRecordAddFingerView Function

Adds an empty finger view to the end of FMRecord finger views.

```
NResult N_API FMRecordAddFingerView(
        HFMRecord hRecord,
        HFmrFingerView * pHFingerView
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pHFingerView	[out] Pointer to a HFmrFingerView that receives handle to created and added FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pHFingerView* is NULL.
N_E_INVALID_OPERATION	Number of finger views in FMRecord (see FMRecordGetFingerViewCount) is equal to FMR_MAX_FINGER_VIEW_COUNT.

See Also

FMRecord Module | HFMRecord | HFmrFingerView | FMRecordGetFingerViewCount

4.2.1.2. FMRecordAddFingerViewCopy Function

Adds a copy of the specified finger view to the end of FMRecord finger views.

```
NResult N_API FMRecordAddFingerViewCopy(
        HFMRecord hRecord,
        HFmrFingerView hSrcFingerView,
        HFmrFingerView * pHFingerView
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
hSrcFingerView	[in] Handle to the FmrFingerView object that is source for added object.
pHFingerView	[out] Pointer to a HFmrFingerView that re-

	ceives handle to created and added FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord*, *hSrcFingerView* or *pHFingerView* is NULL.
N_E_INVALID_OPERATION	Number of finger views in FMRecord (see FMRecordGetFingerViewCount) is equal to FMR_MAX_FINGER_VIEW_COUNT.

See Also

FMRecord Module | HFMRecord | HFmrFingerView | FMRecordGetFingerViewCount

4.2.1.3. FMRecordCalculateSize Function

Calculates the size of a stored FMRecord containing the specified number of finger views of specified size.

```
NResult N_API FMRecordCalculateSize(
        NInt fingerViewCount,
        NSizeType * arFingerViewSizes,
        NSizeType * pSize
);
```

Parameters

fingerViewCount	[in] The number of finger views.
arFingerViewSizes	[in] Pointer to array of NSizeType that contains stored sizes of each finger view.
pSize	[out] Pointer to a NSizeType that receives calculated size of stored FMRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Any entry of array *arFingerViewSizes* points to is less than minimal finger view size.
N_E_ARGUMENT_NULL	*arFingerViewSizes* or *pSize* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*fingerViewCount* is less than zero or greater than FMR_MAX_FINGER_VIEW_COUNT.

Remarks

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

Use `FmrFingerViewGetMaxSize` function to calculate stored size of an individual finger view.

See Also

FMRecord Module | `FMRecordSaveToMemory` | `FmrFingerViewGetMaxSize`

4.2.1.4. FMRecordClearFingerViews Function

Removes all finger views from the FMRecord.

```
NResult N_API FMRecordClearFingerViews(
        HFMRecord hRecord
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

FMRecord Module | HFMRecord

4.2.1.5. FMRecordClone Function

Creates a copy of the FMRecord.

```
NResult N_API FMRecordClone(
        HFMRecord hRecord,
        HFMRecord * pHClonedRecord
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pHClonedRecord	[out] Pointer to a HFMRecord that receives handle to newly created FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hNFRecord* or *pHClonedRecord* is NULL.

Remarks

Created object must be deleted using `FMRecordFree` function.

See Also

FMRecord Module | HFMRecord | `FMRecordFree`

Copyright © 2006 Neurotechnologija

4.2.1.6. FMRecordCreate Function

Creates an empty FMRecord.

```
NResult N_API FMRecordCreate(
        NUShort sizeX,
        NUShort sizeY,
        NUShort resolutionX,
        NUShort resolutionY,
        HFMRecord * pHRecord
);
```

Parameters

sizeX	[in] Specifies horizontal size of fingerprint image.
sizeY	[in] Specifies vertical size of fingerprint image.
resolutionX	[in] Specifies horizontal resolution of fingerprint image.
resolutionY	[in] Specifies vertical resolution of fingerprint image.
pHRecord	[out] Pointer to a HFMRecord that receives handle to newly created FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*sizeX* or *sizeY* is zero. - or - *resolutionX* or *resolutionY* is zero.
N_E_ARGUMENT_NULL	*pHRecord* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `FMRecordFree` function.

See Also

FMRecord Module | HFMRecord | `FMRecordFree`

4.2.1.7. FMRecordCreateFromMemory Function

Restores a FMRecord from the specified memory buffer.

```
NResult N_API FMRecordCreateFromMemory(
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        FMRecordInfo * pInfo,
        HFMRecord * pHRecord
);
```

Parameters

buffer	[in] Pointer to a memory buffer that contains stored FMRecord.
bufferSize	[in] Size of memory buffer that contains stored FMRecord.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pInfo	[out] For internal use. Must be NULL.
pHRecord	[out] Pointer to a HFMRecord that receives handle to newly created FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHRecord* or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is

Error Code	Condition
	inconsistent with FMRecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- FMR_PROCESS_FIRST_FINGER_VIEW_ONLY
- FMRFV_PROCESS_ALL_EXTENDED_DATA
- FMRFV_SKIP_NEUROTEC_FIELDS
- FMRFV_SKIP_RIDGE_COUNTS
- FMRFV_SKIP_SINGULAR_POINTS
- FMRFV_USE_NEUROTEC_FIELDS

Created object must be deleted using `FMRecordFree` function.

See Also

FMRecord Module | HFMRecord | FMRecordInfo | `FMRecordFree` | `FMRecord-SaveToMemory`

4.2.1.8. FMRecordCreateFromNFRecord Function

Creates a FMRecord from the specified NFRecord.

```
NResult N_API FMRecordCreateFromNFRecord(
        HNFRecord hNFRecord,
        NUInt flags,
        HFMRecord * pHRecord
);
```

Parameters

hNFRecord	[in] Handle to the NFRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHRecord	[out] Pointer to a HFMRecord that receives handle to newly created FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hNFRecord* or *pHRecord* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

Created object must be deleted using `FMRecordFree` function.

See Also

FMRecord Module | HFMRecord | `FMRecordFree` | HNFRecord | `FmrFingerView-ToNFRecord`

4.2.1.9. FMRecordCreateFromNFTemplate Function

Creates a FMRecord from the specified NFTemplate.

```
NResult N_API FMRecordCreateFromNFTemplate(
        HNFTemplate hNFTemplate,
        NUInt flags,
        HFMRecord * pHRecord
);
```

Parameters

hNFTemplate	[in] Handle to the NFTemplate object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHRecord	[out] Pointer to a HFMRecord that receives handle to newly created FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hNFTemplate* or *pHRecord* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

Created object must be deleted using `FMRecordFree` function.

See Also

FMRecord Module | HFMRecord | `FMRecordFree` | HNFTemplate | `FMRecordToNFTemplate`

4.2.1.10. FMRecordFree Function

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

```
void N_API FMRecordFree(
        HFMRecord hRecord
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.

Remarks

If *hRecord* is NULL, does nothing.

See Also

FMRecord Module | HFMRecord

4.2.1.11. FMRecordGetCaptureEquipmentCompliance Function

Retrieves the Capture Equipment Compliance of the FMRecord.

```
NResult N_API FMRecordGetCaptureEquipmentCompliance(
        HFMRecord hRecord,
        NByte * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NByte that receives Capture Equipment Compliance 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	*hRecord* or *pValue* is NULL.

See Also

FMRecord Module | HFMRecord | FMRecordSetCaptureEquipmentCompliance

4.2.1.12. FMRecordGetCaptureEquipmentId Function

Retrieves the Capture Equipment Id from of the FMRecord.

```
NResult N_API FMRecordGetCaptureEquipmentId(
        HFMRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NUShort that receives Capture Equipment Id 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	*hRecord* or *pValue* is NULL.

See Also

FMRecord Module | HFMRecord | FMRecordSetCaptureEquipmentId

4.2.1.13. FMRecordGetCbeffProductId Function

Retrieves the CBEFF Product Id of the FMRecord.

```
NResult N_API FMRecordGetCbeffProductId(
        HFMRecord hRecord,
        NUInt * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NUInt that receives CBEFF Product Id 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	*hRecord* or *pValue* is NULL.

See Also

FMRecord Module | HFMRecord | FMRecordSetCbeffProductId

4.2.1.14. FMRecordGetFingerView Function

Retrieves the finger view at the specified index of the FMRecord.

```
NResult N_API FMRecordGetFingerView(
        HFMRecord hRecord,
        NInt index,
        HFmrFingerView * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
index	[in] Index of finger view to retrieve.
pValue	[out] Pointer to a HFmrFingerView that receives handle to FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to finger view count obtained using `FMRecordGetFingerViewCount` function.

See Also

FMRecord Module | HFMRecord | `FMRecordGetFingerViewCount`

4.2.1.15. FMRecordGetFingerViewCapacity Function

Retrieves the number of finger views that the FMRecord can contain.

```
NResult N_API FMRecordGetFingerViewCapacity(
        HFMRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to a NInt that receives number of finger views FMRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Finger view capacity is the number of finger views that the FMRecord can store. Finger view count (see `FMRecordGetFingerViewCount` function) is the number of finger views that are actually in the FMRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger views the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FMRecord Module | HFMRecord | `FMRecordSetFingerViewCapacity` | `FMRecordGetFingerViewCount`

4.2.1.16. FMRecordGetFingerViewCount Function

Retrieves the number of finger views in the FMRecord.

```
NResult N_API FMRecordGetFingerViewCount(
        HFMRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to a NInt that receives number of finger views.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Finger view capacity (see `FMRecordGetFingerViewCapacity` and `FMRecordSetFingerViewCapacity` functions) is the number of finger views that the FMRecord can store. Finger view count is the number of finger views that are actually in the FMRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger views the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FMRecord Module | HFMRecord | `FMRecordGetFingerViewCapacity` | `FMRecordSetFingerViewCapacity`

4.2.1.17. FMRecordGetResolutionX Function

Retrieves horizontal resolution of the image the FMRecord is made from.

```
NResult N_API FMRecordGetResolutionX(
        HFMRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NUShort that receives horizontal resolution of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

FMRecord Module | HFMRecord

4.2.1.18. FMRecordGetResolutionY Function

Retrieves vertical resolution of the image the FMRecord is made from.

```
NResult N_API FMRecordGetResolutionY(
        HFMRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NUShort that receives vertical resolution of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

FMRecord Module | HFMRecord

4.2.1.19. FMRecordGetSize Function

Calculates stored size of the FMRecord.

```
NResult N_API FMRecordGetSize(
        HFMRecord hRecord,
        NUInt flags,
```

```
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to a NSizeType that receives calculated size of stored FMRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL.

Remarks

For the list of flags that are supported see `NFRecordSaveToMemory` function.

See Also

FMRecord Module | HFMRecord | `FMRecordSaveToMemory`

4.2.1.20. FMRecordGetSizeX Function

Retrieves horizontal size of the image the FMRecord is made from.

```
NResult N_API FMRecordGetSizeX(
        HFMRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NUShort that receives horizontal size of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

FMRecord Module | HFMRecord

4.2.1.21. FMRecordGetSizeY Function

Retrieves vertical size of the image the FMRecord is made from.

```
NResult N_API FMRecordGetSizeY(
        HFMRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NUShort that receives vertical size of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

FMRecord Module | HFMRecord

4.2.1.22. FMRecordRemoveFingerView Function

Removes the finger view at the specified index of the FMRecord.

```
NResult N_API FMRecordRemoveFingerView(
        HFMRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
index	[in] Index of finger view to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to finger view count obtained using `FMRecordGetFingerViewCount` function.

See Also

FMRecord Module | HFMRecord | `FMRecordGetFingerViewCount`

4.2.1.23. FMRecordSaveToMemory Function

Stores the FMRecord in the specified memory buffer.

```
NResult N_API FMRecordSaveToMemory(
        HFMRecord hRecord,
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
buffer	[out] Pointer to a memory buffer to store FMRecord. Can be NULL.
bufferSize	[in] Size of memory buffer to store FMRecord. Can be zero.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to a NSizeType that receives size of stored FMRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store FMRecord.
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not zero.

Remarks

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as `FMRecordGetSize` function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with `FMRecordGetSize` function.

The following flags are supported:

- FMR_PROCESS_FIRST_FINGER_VIEW_ONLY
- FMRFV_SKIP_NEUROTEC_FIELDS
- FMRFV_SKIP_RIDGE_COUNTS

Copyright © 2006 Neurotechnologija

- FMRFV_SKIP_SINGULAR_POINTS
- FMRFV_USE_NEUROTEC_FIELDS

See Also

FMRecord Module | HFMRecord | `FMRecordCreateFromMemory` | `FMRecordGetSize`

4.2.1.24. FMRecordSetCaptureEquipmentCompliance Function

Sets the Capture Equipment Compliance of the FMRecord.

```
NResult N_API FMRecordSetCaptureEquipmentCompliance(
        HFMRecord hRecord,
        NByte value
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
value	[in] New Capture Equipment Compliance 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	*hRecord* is NULL.

See Also

FMRecord Module | HFMRecord | `FMRecordGetCaptureEquipmentCompliance`

4.2.1.25. FMRecordSetCaptureEquipmentId Function

Sets the Capture Equipment Id of the FMRecord.

```
NResult N_API FMRecordSetCaptureEquipmentId(
        HFMRecord hRecord,
        NUShort value
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
value	[in] New Capture Equipment Id 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	*hRecord* is NULL.

See Also

FMRecord Module | HFMRecord | FMRecordGetCaptureEquipmentId

4.2.1.26. FMRecordSetCbeffProductId Function

Sets the CBEFF Product Id of the FMRecord.

```
NResult N_API FMRecordSetCbeffProductId(
        HFMRecord hRecord,
        NUInt value
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
value	[in] New CBEFF Product Id 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	*hRecord* is NULL.

See Also

FMRecord Module | HFMRecord | `FMRecordGetCbeffProductId`

4.2.1.27. FMRecordSetFingerViewCapacity Function

Sets the number of finger views that the FMRecord can contain.

```
NResult N_API FMRecordSetFingerViewCapacity(
        HFMRecord hRecord,
        NInt value
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
value	[in] New number of finger views FMRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than finger view count obtained using `FMRecordGetFingerViewCount` function.

Remarks

Finger view capacity is the number of finger views that the FMRecord can store. Finger view count (see `FMRecordGetFingerViewCount` function) is the number of finger views that are actually in the FMRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger views the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

Copyright © 2006 Neurotechnologija

FMRecord Module | HFMRecord | FMRecordGetFingerViewCapacity | FMRecordGetFingerViewCount

4.2.1.28. FMRecordToNFTemplate Function

Converts the FMRecord to a NFTemplate.

```
NResult N_API FMRecordToNFTemplate(
        HFMRecord hRecord,
        NUInt flags,
        HNFTemplate * pHNFTemplate
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHNFTemplate	[out] Pointer to a HNFTemplate that receives handle to newly created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pHNFTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

See Also

FMRecord Module | HFMRecord | HNFTemplate | FMRecordCreateFromNFTemplate

Copyright © 2006 Neurotechnologija

4.2.1.29. FMRecordToNTemplate Function

Converts the FMRecord to a NTemplate.

```
NResult N_API FMRecordToNTemplate(
        HFMRecord hRecord,
        NUInt flags,
        HNTemplate * pHNTemplate
);
```

Parameters

hRecord	[in] Handle to the FMRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHNTemplate	[out] Pointer to a HNTemplate that receives handle to newly created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pHNTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

See Also

FMRecord Module | HFMRecord | HNTemplate

4.2.2. FmrFingerView Module

Provides functionality for editing and converting ANSI INCITS 378-2004 standard templates

finger views (FmrFingerViews) to Neurotechnologija Finger Records (NFRecords).

Header file: `FmrFingerView.h`.

Functions

`FmrFingerViewAddCore`	Adds a FmrCore to the end of FmrFingerView cores.
`FmrFingerViewAddDelta`	Adds a FmrDelta to the end of FmrFingerView deltas.
`FmrFingerViewAddMinutia`	Adds a FmrMinutia to the end of FmrFingerView minutiae.
`FmrFingerViewClearCores`	Removes all cores from the FmrFingerView.
`FmrFingerViewClearDeltas`	Removes all deltas from the FmrFingerView.
`FmrFingerViewClearMinutiae`	Removes all minutiae from the FmrFingerView.
`FmrFingerViewGetCore`	Retrieves the core at the specified index of the FmrFingerView.
`FmrFingerViewGetCoreCapacity`	Retrieves the number of cores that the FmrFingerView can contain.
`FmrFingerViewGetCoreCount`	Retrieves the number of cores in the FmrFingerView.
`FmrFingerViewGetCores`	Copies all cores of FmrFingerView to the specified array.
`FmrFingerViewGetDelta`	Retrieves the delta at the specified index of the FmrFingerView.
`FmrFingerViewGetDeltaCapacity`	Retrieves the number of deltas that the FmrFingerView can contain.
`FmrFingerViewGetDeltaCount`	Retrieves the number of deltas in the FmrFingerView.
`FmrFingerViewGetDeltas`	Copies all deltas of FmrFingerView to the specified array.
`FmrFingerViewGetFingerPosition`	Retrieves the finger position of the FmrFingerView.
`FmrFingerViewGetFingerQuality`	Retrieves the finger quality of the FmrFingerView.

Copyright © 2006 Neurotechnologija

FmrFingerViewGetImpression-Type	Retrieves the impression type of the FmrFingerView.
FmrFingerViewGetMaxSize	Retrieves the maximal size of a FmrFingerView in stored FMRecord containing the specified number of minutiae, cores and deltas and the specified ridge counts.
FmrFingerViewGetMaxSizeEx	For internal use.
FmrFingerViewGetMinutia	Retrieves the minutia at the specified index of the FmrFingerView.
FmrFingerViewGetMinutiaCapa-city	Retrieves the number of minutiae that the FmrFingerView can contain.
FmrFingerViewGetMinutiaCount	Retrieves the number of minutiae in the FmrFingerView.
FmrFingerViewGetMinutiaEight-Neighbour	Retrieves one of the eight minutiae neighbours at the specified index of the minutia at the specified index of the FmrFingerView.
FmrFingerViewGetMinutiaEight-Neighbours	Copies all eight minutia neighbours of the minutia at the specified index of the FmrFingerView to the specified array.
FmrFingerViewGetMinu-tiaFourNeighbour	Retrieves one of the four minutiae neighbours at the specified index of the minutia at the specified index of the FmrFingerView.
FmrFingerViewGetMinu-tiaFourNeighbours	Copies all four minutia neighbours of the minutia at the specified index of the FmrFingerView to the specified array.
FmrFingerViewGetMinutiae	Copies all minutiae of FmrFingerView to the specified array.
FmrFingerViewGetViewNumber	Retrieves the view number of the FmrFingerView.
FmrFingerViewHasEightNeigh-bourRidgeCounts	Retrieves a value indicating whether the FmrFingerView has ridge counts to eight neighbours of each minutia.
FmrFingerViewHasFourNeigh-bourRidgeCounts	Retrieves a value indicating whether the FmrFingerView has ridge counts to four neighbours of each minutia.
FmrFingerViewInsertCore	Inserts a FmrCore into the FmrFingerView at the specified index.

Copyright © 2006 Neurotechnologija

FmrFingerViewInsertDelta	Inserts a FmrDelta into the FmrFingerView at the specified index.
FmrFingerViewInsertMinutia	Inserts a FmrMinutia into the FmrFingerView at the specified index.
FmrFingerViewRemoveCore	Removes the core at the specified index of the FmrFingerView.
FmrFingerViewRemoveDelta	Removes the delta at the specified index of the FmrFingerView.
FmrFingerViewRemoveMinutia	Removes the minutia at the specified index of the FmrFingerView.
FmrFingerViewSetCore	Sets a FmrCore at the specified index of the FmrFingerView.
FmrFingerViewSetCoreCapacity	Sets the number of cores that the FmrFingerView can contain.
FmrFingerViewSetDelta	Sets a FmrDelta at the specified index of the FmrFingerView.
FmrFingerViewSetDeltaCapacity	Sets the number of deltas that the FmrFingerView can contain.
FmrFingerViewSetFingerPosition	Sets the finger position of the FmrFingerView.
FmrFingerViewSetFingerQuality	Sets the finger quality of the FmrFingerView.
FmrFingerViewSetHasEightNeighbourRidgeCounts	Sets a value indicating whether the FmrFingerView has ridge counts to eight neighbours of each minutia.
FmrFingerViewSetHasFourNeighbourRidgeCounts	Sets a value indicating whether the FmrFingerView has ridge counts to four neighbours of each minutia.
FmrFingerViewSetImpressionType	Sets the impression type of the FmrFingerView.
FmrFingerViewSetMinutia	Sets a FmrMinutia at the specified index of the FmrFingerView.
FmrFingerViewSetMinutiaCapacity	Sets the number of minutiae that the FmrFingerView can contain.
FmrFingerViewSetMinutiaEightNeighbour	Sets one of the eight NFMinutiaNeighbours at the specified index of the minutia at the

Copyright © 2006 Neurotechnologija

	specified index of the FmrFingerView.
FmrFingerViewSetMinutiaFourNeighbour	Sets one of the four NFMinutiaNeighbours at the specified index of the minutia at the specified index of the FmrFingerView.
FmrFingerViewSetViewNumber	Sets the view number of the FmrFingerView.
FmrFingerViewToNFRecord	Converts the FmrFingerView to a NFRecord.

Structures

FmrCore	Represents a core in a FmrFingerView.
FmrDelta	Represents a delta in a FmrFingerView.
FmrMinutia	Represents a minutia in a FmrFingerView.

Types

HFmrFingerView	Handle to a FmrFingerView object.

Macros

FMRFV_MAX_CORE_COUNT	The maximum number of cores a FmrFingerView can contain.
FMRFV_MAX_DELTA_COUNT	The maximum number of deltas a FmrFingerView can contain.
FMRFV_MAX_DIMENSION	The maximum value for x and y coordinates of a minutia, core or delta in a FmrFingerView.
FMRFV_MAX_MINUTIA_COUNT	The maximum number of minutiae a FmrFingerView can contain.
FMRFV_NIST_RIDGE_COUNTS	For internal use.
FMRFV_PROCESS_ALL_EXTENDED_DATA	For internal use.
FMRFV_SKIP_NEUROTEC_FIELDS	The flag indicating whether fields of Neurotechnologija should be skipped while load-

	ing or saving FmrFingerView.
FMRFV_SKIP_RIDGE_COUNTS	The flag indicating whether ridge counts should be skipped while loading or saving FmrFingerView.
FMRFV_SKIP_SINGULAR_POINTS	The flag indicating whether singular points (cores and deltas) should be skipped while loading or saving FmrFingerView.
FMRFV_USE_NEUROTEC_FIELDS	The flag indicating whether fields of Neurotechnologija should be used while loading or saving FmrFingerView.

See Also

FMRecord Library

4.2.2.1. FmrCore Structure

Represents a core in a FmrFingerView.

```
typedef struct FmrCore_ { } FmrCore;
```

Fields

Angle	Angle of this FmrCore.
X	X coordinate of this FmrCore.
Y	Y coordinate of this FmrCore.

See Also

FmrFingerView Module

4.2.2.1.1. FmrCore.Angle Field

Angle of this FmrCore.

```
NInt Angle;
```

Remarks

The angle of the core is specified in 2 degrees units in counterclockwise order and can not be

Copyright © 2006 Neurotechnologija

less than zero or greater than 180 minus one.

The value of -1 can be specified if the angle of the core is unknown.

See Also

FmrCore

4.2.2.1.2. FmrCore.X Field

X coordinate of this FmrCore.

```
NUShort X;
```

Remarks

The x coordinate of the core is specified in pixels at x-resolution of FMRecord and can not be greater than FMRFV_MAX_DIMENSION or x-size of FMRecord minus one.

See Also

FmrCore

4.2.2.1.3. FmrCore.Y Field

Y coordinate of this FmrCore.

```
NUShort Y;
```

Remarks

The y coordinate of the core is specified in pixels at y-resolution of FMRecord and can not be greater than FMRFV_MAX_DIMENSION or y-size of FMRecord minus one.

See Also

FmrCore

4.2.2.2. FmrDelta Structure

Represents a delta in a FmrFingerView.

```
typedef struct FmrDelta_ { } FmrDelta;
```

Fields

Angle1	First angle of this FmrDelta.

Angle2	Second angle of this FmrDelta.
Angle3	Third angle of this FmrDelta.
X	X coordinate of this FmrDelta.
Y	Y coordinate of this FmrDelta.

See Also

FmrFingerView Module

4.2.2.2.1. FmrDelta.Angle1 Field

First angle of this FmrDelta.

```
NInt Angle1;
```

Remarks

The angle of the delta is specified in 2 degrees units in counterclockwise order and can not be less than zero or greater than 180 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

See Also

FmrDelta

4.2.2.2.2. FmrDelta.Angle2 Field

Second angle of this FmrDelta.

```
NInt Angle2;
```

Remarks

The angle of the delta is specified in 2 degrees units in counterclockwise order and can not be less than zero or greater than 180 minus one.

The value of -1 can be specified if the second angle of the delta is unknown.

See Also

FmrDelta

4.2.2.2.3. FmrDelta.Angle3 Field

Third angle of this FmrDelta.

```
NInt Angle3;
```

Remarks

The angle of the delta is specified in 2 degrees units in counterclockwise order and can not be less than zero or greater than 180 minus one.

The value of -1 can be specified if the third angle of the delta is unknown.

See Also

FmrDelta

4.2.2.2.4. FmrDelta.X Field

X coordinate of this FmrDelta.

```
NUShort X;
```

Remarks

The x coordinate of the delta is specified in pixels at x-resolution of FMRecord and can not be greater than FMRFV_MAX_DIMENSION or x-size of FMRecord minus one.

See Also

FmrDelta

4.2.2.2.5. FmrDelta.Y Field

Y coordinate of this FmrDelta.

```
NUShort Y;
```

Remarks

The y coordinate of the delta is specified in pixels at y-resolution of FMRecord and can not be greater than FMRFV_MAX_DIMENSION or y-size of FMRecord minus one.

See Also

FmrDelta

4.2.2.3. FmrFingerViewAddCore Function

Adds a FmrCore to the end of FmrFingerView cores.

Copyright © 2006 Neurotechnologija

```
NResult N_API FmrFingerViewAddCore(
        HFmrFingerView hFingerView,
        const FmrCore * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[in] Pointer to the FmrCore to add.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_INVALID_OPERATION	Number of cores in FmrFingerView (see FmrFingerViewGetCoreCount) is equal to FMRFV_MAX_CORE_COUNT.

See Also

FmrFingerView Module | HFmrFingerView | FmrCore

4.2.2.4. FmrFingerViewAddDelta Function

Adds a FmrDelta to the end of FmrFingerView deltas.

```
NResult N_API FmrFingerViewAddDelta(
        HFmrFingerView hFingerView,
        const FmrDelta * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[in] Pointer to the FmrDelta to add.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_INVALID_OPERATION	Number of deltas in FmrFingerView (see FmrFingerViewGetDeltaCount) is equal to FMRFV_MAX_DELTA_COUNT.

See Also

FmrFingerView Module | HFmrFingerView | FmrDelta

4.2.2.5. FmrFingerViewAddMinutia Function

Adds a FmrMinutia to the end of FmrFingerView minutiae.

```
NResult N_API FmrFingerViewAddMinutia(
        HFmrFingerView hFingerView,
        const FmrMinutia * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[in] Pointer to the FmrMinutia to add.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

Error Code	Condition
N_E_INVALID_OPERATION	Number of minutiae in FmrFingerView (see `FmrFingerViewGetMinutiaCount`) is equal to FMR-FV_MAX_MINUTIA_COUNT.

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia

4.2.2.6. FmrFingerViewClearCores Function

Removes all cores from the FmrFingerView.

```
NResult N_API FmrFingerViewClearCores(
        HFmrFingerView hFingerView
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView

4.2.2.7. FmrFingerViewClearDeltas Function

Removes all deltas from the FmrFingerView.

```
NResult N_API FmrFingerViewClearDeltas(
        HFmrFingerView hFingerView
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView

4.2.2.8. FmrFingerViewClearMinutiae Function

Removes all minutiae from the FmrFingerView.

```
NResult N_API FmrFingerViewClearMinutiae(
        HFmrFingerView hFingerView
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView

4.2.2.9. FmrFingerViewGetCore Function

Retrieves the core at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewGetCore(
        HFmrFingerView hFingerView,
        NInt index,
        FmrCore * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index of core to retrieve.
pValue	[out] Pointer to FmrCore that receives core.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to core count obtained using FmrFingerViewGetCoreCount function.

See Also

FmrFingerView Module | HFmrFingerView | FmrCore | FmrFingerViewGetCoreCount | FmrFingerViewSetCore

4.2.2.10. FmrFingerViewGetCoreCapacity Function

Retrieves the number of cores that the FmrFingerView can contain.

```
NResult N_API FmrFingerViewGetCoreCapacity(
        HFmrFingerView hFingerView,
        NInt * pValue
);
```

Copyright © 2006 Neurotechnologija

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NInt that receives number of cores FmrFingerView can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

Remarks

Core capacity is the number of cores that the FmrFingerView can store. Core count (see `FmrFingerViewGetCoreCount` function) is the number of cores that are actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewSetCoreCapacity` | `FmrFingerViewGetCoreCount`

4.2.2.11. FmrFingerViewGetCoreCount Function

Retrieves the number of cores in the FmrFingerView.

```
NResult N_API FmrFingerViewGetCoreCount(
        HFmrFingerView hFingerView,
        NInt * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NInt that receives number of

	cores.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

Remarks

Core capacity (see `FmrFingerViewGetCoreCapacity` and `FmrFingerViewSetCoreCapacity` functions) is the number of cores that the FmrFingerView can store. Core count is the number of cores that are actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewGetCoreCapacity` | `FmrFingerViewSetCoreCapacity`

4.2.2.12. FmrFingerViewGetCores Function

Copies all cores of FmrFingerView to the specified array.

```
NResult N_API FmrFingerViewGetCores(
        HFmrFingerView hFingerView,
        FmrCore * arCores
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
arCores	[out] Pointer to array of FmrCore that receives cores.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *arCores* is NULL.

Remarks

Array *arCores* points to must be large enough to receive all FmrFingerView cores. See FmrFingerViewGetCoreCount function.

See Also

FmrFingerView Module | HFmrFingerView | FmrCore | FmrFingerViewGetCore-Count

4.2.2.13. FmrFingerViewGetDelta Function

Retrieves the delta at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewGetDelta(
        HFmrFingerView hFingerView,
        NInt index,
        FmrDelta * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index of delta to retrieve.
pValue	[out] Pointer to FmrDelta that receives delta.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or

Error Code	Condition
	equal to delta count obtained using `FmrFingerViewGetDeltaCount` function.

See Also

FmrFingerView Module | HFmrFingerView | FmrDelta | `FmrFingerViewGetDeltaCount` | `FmrFingerViewSetDelta`

4.2.2.14. FmrFingerViewGetDeltaCapacity Function

Retrieves the number of deltas that the FmrFingerView can contain.

```
NResult N_API FmrFingerViewGetDeltaCapacity(
        HFmrFingerView hFingerView,
        NInt * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NInt that receives number of deltas FmrFingerView can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

Remarks

Delta capacity is the number of deltas that the FmrFingerView can store. Delta count (see `FmrFingerViewGetDeltaCount` function) is the number of deltas that are actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding

deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewSetDeltaCapacity` | `FmrFingerViewGetDeltaCount`

4.2.2.15. FmrFingerViewGetDeltaCount Function

Retrieves the number of deltas in the FmrFingerView.

```
NResult N_API FmrFingerViewGetDeltaCount(
        HFmrFingerView hFingerView,
        NInt * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NInt that receives number of deltas.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

Remarks

Delta capacity (see `FmrFingerViewGetDeltaCapacity` and `FmrFingerViewSetDeltaCapacity` functions) is the number of deltas that the FmrFingerView can store. Delta count is the number of deltas that are actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

Copyright © 2006 Neurotechnologija

FmrFingerView Module | HFmrFingerView | `FmrFingerViewGetDeltaCapacity` | `FmrFingerViewSetDeltaCapacity`

4.2.2.16. FmrFingerViewGetDeltas Function

Copies all deltas of FmrFingerView to the specified array.

```
NResult N_API FmrFingerViewGetDeltas(
        HFmrFingerView hFingerView,
        FmrDelta * arDeltas
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
arDeltas	[out] Pointer to array of FmrDelta that receives deltas.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *arDeltas* is NULL.

Remarks

Array *arDeltas* points to must be large enough to receive all FmrFingerView deltas. See `FmrFingerViewGetDeltaCount` function.

See Also

FmrFingerView Module | HFmrFingerView | FmrDelta | `FmrFingerViewGetDeltaCount`

4.2.2.17. FmrFingerViewGetFingerPosition Function

Retrieves the finger position of the FmrFingerView.

```
NResult N_API FmrFingerViewGetFingerPosition(
        HFmrFingerView hFingerView,
        NFPosition * pValue
```

```
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NFPosition that receives finger position.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pFingerView* or *pValue* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | NFPosition | `FmrFingerViewSetFingerPosition`

4.2.2.18. FmrFingerViewGetFingerQuality Function

Retrieves the finger quality of the FmrFingerView.

```
NResult N_API FmrFingerViewGetFingerQuality(
        HFmrFingerView hFingerView,
        NByte * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NByte that receives finger quality.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetFingerQuality

4.2.2.19. FmrFingerViewGetImpressionType Function

Retrieves the impression type of the FmrFingerView.

```
NResult N_API FmrFingerViewGetImpressionType(
        HFmrFingerView hFingerView,
        NFImpressionType * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NFImpressionType that receives impression type.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | NFImpressionType | FmrFingerViewSetImpressionType

4.2.2.20. FmrFingerViewGetMaxSize Function

Retrieves the maximal size of a FmrFingerView in stored FMRecord containing the specified

Copyright © 2006 Neurotechnologija

number of minutiae, cores and deltas and the specified ridge counts.

```
NResult N_API FmrFingerViewGetMaxSize(
        NInt minutiaCount,
        NBool hasFourNeighbourRidgeCounts,
        NBool hasEightNeighbourRidgeCounts,
        NBool coresHasAngles,
        NInt coreCount,
        NBool deltasHasAngles,
        NInt deltaCount,
        NSizeType * pSize
);
```

Parameters

minutiaCount	[in] The number of minutiae.
hasFourNeighbourRidgeCounts	[in] NTrue if finger view has ridge counts to four neighbours of each minutia. NFalse otherwise.
hasEightNeighbourRidgeCounts	[in] NTrue if finger view has ridge counts to eight neighbours of each minutia. NFalse otherwise.
coresHasAngles	[in] NTrue if finger view cores has angles. NFalse otherwise.
coreCount	[in] The number of cores.
deltasHasAngles	[in] NTrue if finger view deltas has angles. NFalse otherwise.
deltaCount	[in] The number of deltas.
pSize	[out] Pointer to NSizeType that receives the maximal size of stored FmrFingerView.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pSize* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaCount* is less than zero or greater than FMRFV_MAX_MINUTIA_COUNT.

Copyright © 2006 Neurotechnologija

Error Code	Condition
	- or - *coreCount* is less than zero or greater than FMRFV_MAX_CORE_COUNT. - or - *deltaCount* is less than zero or greater than FMRFV_MAX_DELTA_COUNT.

Remarks

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

See Also

FmrFingerView Module

4.2.2.21. FmrFingerViewGetMinutia Function

Retrieves the minutia at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewGetMinutia(
        HFmrFingerView hFingerView,
        NInt index,
        FmrMinutiaFmrMinutia * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index of minutia to retrieve.
pValue	[out] Pointer to FmrMinutia that receives minutia.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using FmrFingerViewGetMinutiaCount function.

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia | FmrFingerViewGetMinutiaCount | FmrFingerViewSetMinutia

4.2.2.22. FmrFingerViewGetMinutiaCapacity Function

Retrieves the number of minutiae that the FmrFingerView can contain.

```
NResult N_API FmrFingerViewGetMinutiaCapacity(
        HFmrFingerView hFingerView,
        NInt * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NInt that receives number of minutiae FmrFingerView can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

Remarks

Minutia capacity is the number of minutiae that the FmrFingerView can store. Minutia count (see FmrFingerViewGetMinutiaCount function) is the number of minutiae that are

actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewSetMinutiaCapacity` | `FmrFingerViewGetMinutiaCount`

4.2.2.23. FmrFingerViewGetMinutiaCount Function

Retrieves the number of minutiae in the FmrFingerView.

```
NResult N_API FmrFingerViewGetMinutiaCount(
        HFmrFingerView hFingerView,
        NInt * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NInt that receives number of minutiae.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

Remarks

Minutia capacity (see `FmrFingerViewGetMinutiaCapacity` and `FmrFingerViewSetMinutiaCapacity` functions) is the number of minutiae that the FmrFingerView can store. Minutia count is the number of minutiae that are actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

Copyright © 2006 Neurotechnologija

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewGetMinutiaCapacity` | `FmrFingerViewSetMinutiaCapacity`

4.2.2.24. FmrFingerViewGetMinutiaEightNeighbour Function

Retrieves one of the eight minutiae neighbours at the specified index of the minutia at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewGetMinutiaEightNeighbour(
        HFmrFingerView hFingerView,
        NInt minutiaIndex,
        NInt index,
        NFMinutiaNeighbour * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
minutiaIndex	[in] The index of minutia.
index	[in] Index of minutia neighbour to retrieve.
pValue	[out] Pointer to NFMinutiaNeighbour that receives minutia neighbour.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaIndex* is less than zero or greater than or equal to minutia count obtained using `FmrFingerViewGetMinutiaCount` function. - or - *index* is less than zero or greater than or equal to 8.

See Also

FmrFingerView Module | HFmrFingerView | NFMinutiaNeighbour | `FmrFingerView-GetMinutiaCount` | `FmrFingerViewSetMinutiaEightNeighbour`

4.2.2.25. FmrFingerViewGetMinutiaEightNeighbours Function

Copies all eight minutia neighbours of the minutia at the specified index of the FmrFinger-View to the specified array.

```
NResult N_API FmrFingerViewGetMinutiaEightNeighbours(
        HFmrFingerView hFingerView,
        NInt minutiaIndex,
        NFMinutiaNeighbour * arMinutiaNeighbours
);
```

Parameters

`hFingerView`	[in] Handle to the FmrFingerView object.
`minutiaIndex`	[in] The index of minutia.
`arMinutiaNeighbours`	[out] Pointer to array of NFMinutiaNeigh-bour that receives minutia neighbours.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*`hFingerView`* or *`arMinutiaNeigh-bours`* is NULL.

Remarks

Array *`arMinutiaNeighbours`* points to must be large enough to receive all four minutia neighbours.

See Also

FmrFingerView Module | HFmrFingerView | NFMinutiaNeighbour | `FmrFingerView-GetMinutiaCount`

4.2.2.26. FmrFingerViewGetMinutiaFourNeighbour Function

Retrieves one of the four minutiae neighbours at the specified index of the minutia at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewGetMinutiaFourNeighbour(
        HFmrFingerView hFingerView,
        NInt minutiaIndex,
        NInt index,
        NFMinutiaNeighbour * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
minutiaIndex	[in] The index of minutia.
index	[in] Index of minutia neighbour to retrieve.
pValue	[out] Pointer to NFMinutiaNeighbour that receives minutia neighbour.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaIndex* is less than zero or greater than or equal to minutia count obtained using FmrFingerViewGetMinutiaCount function. - or - *index* is less than zero or greater than or equal to 4.

See Also

FmrFingerView Module | HFmrFingerView | NFMinutiaNeighbour | FmrFingerViewGetMinutiaCount | FmrFingerViewSetMinutiaFourNeighbour

Copyright © 2006 Neurotechnologija

4.2.2.27. FmrFingerViewGetMinutiaFourNeighbours Function

Copies all four minutia neighbours of the minutia at the specified index of the FmrFingerView to the specified array.

```
NResult N_API FmrFingerViewGetMinutiaFourNeighbours(
        HFmrFingerView hFingerView,
        NInt minutiaIndex,
        NFMinutiaNeighbour * arMinutiaNeighbours
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
minutiaIndex	[in] The index of minutia.
arMinutiaNeighbours	[out] Pointer to array of NFMinutiaNeighbour that receives minutia neighbours.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *arMinutiaNeighbours*is NULL.

Remarks

Array *arMinutiaNeighbours* points to must be large enough to receive all four minutia neighbours.

See Also

FmrFingerView Module | HFmrFingerView | NFMinutiaNeighbour | FmrFingerViewGetMinutiaCount

4.2.2.28. FmrFingerViewGetMinutiae Function

Copies all minutiae of FmrFingerView to the specified array.

```
NResult N_API FmrFingerViewGetMinutiae(
```

```
        HFmrFingerView hFingerView,
        FmrMinutia * arMinutiae
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
arMinutiae	[out] Pointer to array of FmrMinutia that receives minutiae.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pFingerView* or *arMinutiae* is NULL.

Remarks

Array *arMinutiae* points to must be large enough to receive all FmrFingerView minutiae. See `FmrFingerViewGetMinutiaCount` function.

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia | `FmrFingerViewGetMinutiaCount`

4.2.2.29. FmrFingerViewGetViewNumber Function

Retrieves the view number of the FmrFingerView.

```
NResult N_API FmrFingerViewGetViewNumber(
        HFmrFingerView hFingerView,
        NByte * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NByte that receives view

	number.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetViewNumber

4.2.2.30. FmrFingerViewHasEightNeighbourRidgeCounts Function

Retrieves a value indicating whether the FmrFingerView has ridge counts to eight neighbours of each minutia.

```
NResult N_API FmrFingerViewHasEightNeighbourRidgeCounts(
        HFmrFingerView hFingerView,
        NBool * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NBool that receives value indicating whether FmrFingerView contains ridge counts to eight neighbours of each minutia.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetHasEightNeighbourRidgeCounts

4.2.2.31. FmrFingerViewHasFourNeighbourRidgeCounts Function

Retrieves a value indicating whether the FmrFingerView has ridge counts to four neighbours of each minutia.

```
NResult N_API FmrFingerViewHasFourNeighbourRidgeCounts(
        HFmrFingerView hFingerView,
        NBool * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pValue	[out] Pointer to NBool that receives value indicating whether FmrFingerView contains ridge counts to four neighbours of each minutia.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetHasFourNeighbourRidgeCounts

4.2.2.32. FmrFingerViewInsertCore Function

Inserts a FmrCore into the FmrFingerView at the specified index.

```
NResult N_API FmrFingerViewInsertCore(
        HFmrFingerView hFingerView,
        NInt index,
        const FmrCore * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index at which core is inserted.
pValue	[in] Pointer to the FmrCore to insert.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than core count obtained using `FmrFingerView-GetCoreCount` function.
N_E_INVALID_OPERATION	Number of cores in FmrFingerView (see `FmrFingerViewGetCoreCount`) is equal to FMRFV_MAX_CORE_COUNT.

See Also

FmrFingerView Module | HFmrFingerView | FmrCore | `FmrFingerViewGetCore-Count`

4.2.2.33. FmrFingerViewInsertDelta Function

Inserts a FmrDelta into the FmrFingerView at the specified index.

```
NResult N_API FmrFingerViewInsertDelta(
        HFmrFingerView hFingerView,
        NInt index,
        const FmrDelta * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index at which delta is inserted.
pValue	[in] Pointer to the FmrDelta to insert.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than delta count obtained using `FmrFingerView-GetDeltaCount` function.
N_E_INVALID_OPERATION	Number of deltas in FmrFingerView (see `FmrFingerViewGetDeltaCount`) is equal to FMRFV_MAX_DELTA_COUNT.

See Also

FmrFingerView Module | HFmrFingerView | FmrDelta | `FmrFingerViewGetDelta-Count`

4.2.2.34. FmrFingerViewInsertMinutia Function

Inserts a FmrMinutia into the FmrFingerView at the specified index.

```
NResult N_API FmrFingerViewInsertMinutia(
        HFmrFingerView hFingerView,
        NInt index,
        const FmrMinutia * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index at which minutia is inserted.
pValue	[in] Pointer to the FmrMinutia to insert.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than minutia count obtained using FmrFingerViewGetMinutiaCount function.
N_E_INVALID_OPERATION	Number of minutiae in FmrFingerView (see FmrFingerViewGetMinutiaCount) is equal to FMRFV_MAX_MINUTIA_COUNT.

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia | FmrFingerViewGetMinutiaCount

4.2.2.35. FmrFingerViewRemoveCore Function

Removes the core at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewRemoveCore(
        HFmrFingerView hFingerView,
        NInt index
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

index	[in] Index of core to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to core count obtained using FmrFingerViewGetCoreCount function.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetCoreCount

4.2.2.36. FmrFingerViewRemoveDelta Function

Removes the delta at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewRemoveDelta(
        HFmrFingerView hFingerView,
        NInt index
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index of delta to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to delta count obtained using FmrFingerViewGetDeltaCount function.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetDeltaCount

4.2.2.37. FmrFingerViewRemoveMinutia Function

Removes the minutia at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewRemoveMinutia(
        HFmrFingerView hFingerView,
        NInt index
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index of minutia to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using FmrFingerViewGetMinutiaCount function.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetMinutiaCount

4.2.2.38. FmrFingerViewSetCore Function

Sets a FmrCore at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewSetCore(
        HFmrFingerView hFingerView,
        NInt index,
        const FmrCore * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index of core to set.
pValue	[in] Pointer to the FmrCore to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to core count obtained using FmrFingerViewGetCoreCount function.

See Also

FmrFingerView Module | HFmrFingerView | FmrCore | FmrFingerViewGetCoreCount | FmrFingerViewGetCore

4.2.2.39. FmrFingerViewSetCoreCapacity Function

Sets the number of cores that the FmrFingerView can contain.

```
NResult N_API FmrFingerViewSetCoreCapacity(
        HFmrFingerView hFingerView,
        NInt value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
value	[in] New number of cores FmrFingerView can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than core count obtained using FmrFingerViewGetCoreCount function.

Remarks

Core capacity is the number of cores that the FmrFingerView can store. Core count (see FmrFingerViewGetCoreCount function) is the number of cores that are actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetCoreCapacity | FmrFingerViewGetCoreCount

4.2.2.40. FmrFingerViewSetDelta Function

Sets a FmrDelta at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewSetDelta(
        HFmrFingerView hFingerView,
        NInt index,
        const FmrDelta * value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index of delta to set.
pValue	[in] Pointer to the FmrDelta to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to delta count obtained using `FmrFingerViewGetDeltaCount` function.

See Also

FmrFingerView Module | HFmrFingerView | FmrDelta | `FmrFingerViewGetDeltaCount` | `FmrFingerViewGetDelta`

4.2.2.41. FmrFingerViewSetDeltaCapacity Function

Sets the number of deltas that the FmrFingerView can contain.

```
NResult N_API FmrFingerViewSetDeltaCapacity(
        HFmrFingerView hFingerView,
        NInt value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
value	[in] New number of deltas FmrFingerView can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than delta count obtained using FmrFingerViewGetDeltaCount function.

Remarks

Delta capacity is the number of deltas that the FmrFingerView can store. Delta count (see FmrFingerViewGetDeltaCount function) is the number of deltas that are actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetDeltaCapacity | FmrFingerViewGetDeltaCount

4.2.2.42. FmrFingerViewSetFingerPosition Function

Sets the finger position of the FmrFingerView.

```
NResult N_API FmrFingerViewSetFingerPosition(
        HFmrFingerView hFingerView,
        NFPosition value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
value	[in] New finger position value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | NFPosition | FmrFingerViewGetFinger-Position

4.2.2.43. FmrFingerViewSetFingerQuality Function

Sets the finger quality of the FmrFingerView.

```
NResult N_API FmrFingerViewSetFingerQuality(
        HFmrFingerView hFingerView,
        NByte value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
value	[in] New finger quality value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetFingerQuality

4.2.2.44. FmrFingerViewSetHasEightNeighbourRidgeCounts Function

Sets a value indicating whether the FmrFingerView has ridge counts to eight neighbours of each minutia.

```
NResult N_API FmrFingerViewSetHasEightNeighbourRidgeCounts(
        HFmrFingerView hFingerView,
        NBool value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
value	[in] New value indicating whether FmrFingerView has ridge counts to eight neighbours of each minutia.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewGetHasEightNeighbourRidgeCounts`

4.2.2.45. FmrFingerViewSetHasFourNeighbourRidgeCounts Function

Sets a value indicating whether the FmrFingerView has ridge counts to four neighbours of each minutia.

```
NResult N_API FmrFingerViewSetHasFourNeighbourRidgeCounts(
        HFmrFingerView hFingerView,
        NBool value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

value	[in] New value indicating whether FmrFingerView has ridge counts to four neighbours of each minutia.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetHasFourNeighbourRidgeCounts

4.2.2.46. FmrFingerViewSetImpressionType Function

Sets the impression type of the FmrFingerView.

```
NResult N_API FmrFingerViewSetImpressionType(
        HFmrFingerView hFingerView,
        NFImpressionType value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
value	[in] New impression type value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | NFImpressionType | `FmrFingerViewGetImpressionType`

4.2.2.47. FmrFingerViewSetMinutia Function

Sets a FmrMinutia at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewSetMinutia(
        HFmrFingerView hFingerView,
        NInt index,
        const FmrMinutia * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
index	[in] Index of minutia to set.
pValue	[in] Pointer to the FmrMinutia to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using `FmrFingerViewGetMinutiaCount` function.

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia | `FmrFingerViewGetMinutiaCount` | `FmrFingerViewGetMinutia`

4.2.2.48. FmrFingerViewSetMinutiaCapacity Function

Copyright © 2006 Neurotechnologija

Sets the number of minutiae that the FmrFingerView can contain.

```
NResult N_API FmrFingerViewSetMinutiaCapacity(
        HFmrFingerView hFingerView,
        NInt value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
value	[in] New number of minutiae FmrFingerView can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than minutia count obtained using `FmrFingerViewGetMinutiaCount` function.

Remarks

Minutia capacity is the number of minutiae that the FmrFingerView can store. Minutia count (see `FmrFingerViewGetMinutiaCount` function) is the number of minutiae that are actually in the FmrFingerView.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewGetMinutiaCapacity` | `FmrFingerViewGetMinutiaCount`

4.2.2.49. FmrFingerViewSetMinutiaEightNeighbour Function

Sets one of the eight NFMinutiaNeighbours at the specified index of the minutia at the spe-

cified index of the FmrFingerView.

```
NResult N_API FmrFingerViewSetMinutiaEightNeighbour(
        HFmrFingerView hFingerView,
        NInt minutiaIndex,
        NInt index,
        const NFMinutiaNeighbour * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
minutiaIndex	[in] The index of minutia.
index	[in] Index of minutia neighbour to set.
pValue	[in] Pointer to the NFMinutiaNeighbour to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaIndex* is less than zero or greater than or equal to minutia count obtained using FmrFingerViewGetMinutiaCount function. - or - *index* is less than zero or greater than or equal to 8.

See Also

FmrFingerView Module | HFmrFingerView | NFMinutiaNeighbour | FmrFingerViewGetMinutiaCount | FmrFingerViewGetMinutiaEightNeighbour

4.2.2.50. FmrFingerViewSetMinutiaFourNeighbour Function

Sets one of the four NFMinutiaNeighbours at the specified index of the minutia at the specified index of the FmrFingerView.

```
NResult N_API FmrFingerViewSetMinutiaFourNeighbour(
        HFmrFingerView hFingerView,
        NInt minutiaIndex,
        NInt index,
        const NFMinutiaNeighbour * pValue
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
minutiaIndex	[in] The index of minutia.
index	[in] Index of minutia neighbour to set.
pValue	[in] Pointer to the NFMinutiaNeighbour to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaIndex* is less than zero or greater than or equal to minutia count obtained using FmrFingerViewGetMinutiaCount function. - or - *index* is less than zero or greater than or equal to 4.

See Also

FmrFingerView Module | HFmrFingerView | NFMinutiaNeighbour | FmrFingerViewGetMinutiaCount | FmrFingerViewGetMinutiaFourNeighbour

Copyright © 2006 Neurotechnologija

4.2.2.51. FmrFingerViewSetViewNumber Function

Sets the view number of the FmrFingerView.

```
NResult N_API FmrFingerViewSetViewNumber(
        HFmrFingerView hFingerView,
        NByte value
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
value	[in] New view number 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	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewGetViewNumber`

4.2.2.52. FmrFingerViewToNFRecord Function

Converts the FmrFingerView to a NFRecord.

```
NResult N_API FmrFingerViewToNFRecord(
        HFmrFingerView hFingerView,
        NUInt flags,
        HNFRecord * pHNFRecord
);
```

Parameters

hFingerView	[in] Handle to the FmrFingerView object.
pHNFRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHNFRecord* or *pFingerView* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

See Also

FmrFingerView Module | HFmrFingerView | HNFRecord | FMRecordCreateFromN-FRecord

4.2.2.53. FmrMinutia Structure

Represents a minutia in a FmrFingerView.

```
typedef struct FmrMinutia_ { } FmrMinutia;
```

Fields

Angle	Angle of this FmrMinutia.
Quality	Quality of this FmrMinutia.
Type	Type of this FmrMinutia.
X	X coordinate of this FmrMinutia.
Y	Y coordinate of this FmrMinutia.

See Also

FmrFingerView Module

4.2.2.53.1. FmrMinutia.Angle Field

Angle of this FmrMinutia.

```
NByte Angle;
```

Remarks

The angle of the minutia is specified in 2 degrees units in counterclockwise order and can not be greater than 180 minus one.

See Also

FmrMinutia

4.2.2.53.2. FmrMinutia.Quality Field

Quality of this FmrMinutia.

```
NByte Quality;
```

Remarks

The quality of the minutia must be in the range [0, 100]. The higher it is, the better the quality of the minutia is.

If quality of the minutia is unknown it must be set to zero.

See Also

FmrMinutia

4.2.2.53.3. FmrMinutia.Type Field

Type of this FmrMinutia.

```
NFMinutiaType Type;
```

See Also

FmrMinutia

4.2.2.53.4. FmrMinutia.X Field

X coordinate of this FmrMinutia.

```
NUShort X;
```

Copyright © 2006 Neurotechnologija

Remarks

The x coordinate of the minutia is specified in pixels at x-resolution of FMRecord and can not be greater than FMRFV_MAX_DIMENSION or x-size of FMRecord minus one.

See Also

FmrMinutia

4.2.2.53.5. FmrMinutia.Y Field

Y coordinate of this FmrMinutia.

```
NUShort Y;
```

Remarks

The y coordinate of the minutia is specified in pixels at y-resolution of FMRecord and can not be greater than FMRFV_MAX_DIMENSION or y-size of FMRecord minus one.

See Also

FmrMinutia

4.3. NCore Library

Provides infrastructure for Neurotechnologija components.

Import library (Windows): `NCore.dll.lib`.

DLL (Windows): `NCore.dll`.

Shared object (Linux): `libNCore.so`.

Requirements (Windows):

- `msvcr71.dll` (Microsoft C Runtime Library 7.1).

Modules

NCore	Provides infrastructure/basic functionality (such as memory management) for Neurotechnologija components.
NErrors	Defines error codes used in Neurotechnologija components.
NParameters	Provides functionality for work with para-

	meters for Neurotechnologija components.
NTypes	Defines types and macros used in Neurotechnologija components.

4.3.1. NCore Module

Provides infrastructure/basic functionality (such as memory management) for Neurotechnologija components.

Header file: `NCore.h`.

Functions

`NAlloc`	Allocates memory block.
`NAStrLen`	Gets length of a null-terminated string of ansi chars.
`NCAlloc`	Allocates memory block with all bytes set to zero.
`NCompare`	Compares bytes in two memory blocks.
`NCompareAStr`	Compares two null-terminated strings of ansi chars.
`NCompareStr`	Compares two null-terminated strings of chars.
`NCopy`	Copies data between memory blocks.
`NCopyAStr`	Copies null-terminated string of ansi chars.
`NCopyStr`	Copies null-terminated string of chars.
`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.
`NStrLen`	Gets length of a null-terminated string of chars.

Copyright © 2006 Neurotechnologija

Macros

NClear	Clears memory block.

See Also

NCore Library

4.3.2. 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.
-201	N_E_LM_CONNECTION	Error during connection with LAN License Manager has occurred.
-202	N_E_LM_NO_MORE_LICENSE	Library has not been registered because registered license count achieved LAN License Manager license limit.
-5	N_E_NOT_IMPLEMENTED	Functionality is not implemented.

Copyright © 2006 Neurotechnologija

-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

4.3.3. NParameters Module

Provides functionality for work 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.

Copyright © 2006 Neurotechnologija

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

4.3.3.1. 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

4.3.4. 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 (same as NAChar).
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

Copyright © 2006 Neurotechnologija

	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.

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_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_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_LONG_MAX	Maximum value for NLong.
N_LONG_MIN	Minimum value for NLong.
N_MSVC	Defined if compiling with Microsoft Visual C++.
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_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.

Copyright © 2006 Neurotechnologija

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_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_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

Copyright © 2006 Neurotechnologija

4.3.4.1. 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

4.3.4.2. 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

4.3.4.3. 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

4.3.4.3.1. NIndexPair.Index1 Field

First index of this NIndexPair.

```
NInt Index1;
```

See Also

NIndexPair

4.3.4.3.2. NIndexPair.Index2 Field

Second index of this NIndexPair.

```
NInt Index2;
```

See Also

NIndexPair

4.3.4.4. 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

4.3.4.4.1. NRational.Denominator Field

Denominator of this NRational.

```
NInt Denominator;
```

See Also

NRational

4.3.4.4.2. NRational.Numerator Field

Numerator of this NRational.

```
NInt Numerator;
```

See Also

NRational

4.3.4.5. 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

4.3.4.5.1. NURational.Denominator Field

Denominator of this NURational.

```
NUInt Denominator;
```

See Also

NURational

Copyright © 2006 Neurotechnologija

4.3.4.5.2. NURational.Numerator Field

Numerator of this NURational.

```
NUInt Numerator;
```

See Also

NURational

4.4. NFRecord Library

Provides functionality for packing, unpacking and editing Neurotechnologija Finger Records.

Import library (Windows): `NFRecord.dll.lib`.

DLL (Windows): `NFRecord.dll`.

Shared object (Linux): `libNFRecord.so`.

Requirements (Windows):

- `NCore.dll`.

Requirements (Linux):

- `libNCore.so`.

Modules

NFRecord	Provides functionality for packing, unpacking and editing Neurotechnologija Finger Records (NFRecords).

4.4.1. NFRecord Module

Provides functionality for packing, unpacking and editing Neurotechnologija Finger Records (NFRecords).

Header file: `NFRecord.h` and `NFRecordV1.h`

Functions

`NFRecordAddCore`	Adds a NFCore to the end of NFRecord cores.

NFRecordAddDelta	Adds a NFDelta to the end of NFRecord deltas.
NFRecordAddDoubleCore	Adds a NFDoubleCore to the end of NFRecord double cores.
NFRecordAddMinutia	Adds a NFMinutia to the end of NFRecord minutiae.
NFRecordCheck	Checks if format of the packed NFRecord is correct.
NFRecordClearCores	Removes all cores from the NFRecord.
NFRecordClearDeltas	Removes all deltas from the NFRecord.
NFRecordClearDoubleCores	Removes all double cores from the NFRecord.
NFRecordClearMinutiae	Removes all minutiae from the NFRecord.
NFRecordClone	Creates a copy of the NFRecord.
NFRecordCreate	Creates an empty NFRecord.
NFRecordCreateFromMemory	Unpacks a NFRecord from the specified memory buffer.
NFRecordFree	Deletes the NFRecord. After the object is deleted the specified handle is no longer valid.
NFRecordGetCore	Retrieves the core at the specified index of the NFRecord.
NFRecordGetCoreCapacity	Retrieves the number of cores that the NFRecord can contain.
NFRecordGetCoreCount	Retrieves the number of cores in the NFRecord.
NFRecordGetCores	Copies all cores of NFRecord to the specified array.
NFRecordGetDelta	Retrieves the delta at the specified index of the NFRecord.
NFRecordGetDeltaCapacity	Retrieves the number of deltas that the NFRecord can contain.
NFRecordGetDeltaCount	Retrieves the number of deltas in the NFRecord.

Copyright © 2006 Neurotechnologija

NFRecordGetDeltas	Copies all deltas of NFRecord to the specified array.
NFRecordGetDoubleCore	Retrieves the double core at the specified index of the NFRecord.
NFRecordGetDoubleCoreCapacity	Retrieves the number of double cores that the NFRecord can contain.
NFRecordGetDoubleCoreCount	Retrieves the number of double cores in the NFRecord.
NFRecordGetDoubleCores	Copies all double cores of NFRecord to the specified array.
NFRecordGetG	Retrieves the G of the NFRecord.
NFRecordGetGMem	Retrieves the G of the packed NFRecord.
NFRecordGetHeight	Retrieves the height of the image the NFRecord is made from.
NFRecordGetHeightMem	Retrieves the height of the image the packed NFRecord is made from.
NFRecordGetHorzResolution	Retrieves the horizontal resolution of the image the NFRecord is made from.
NFRecordGetHorzResolutionMem	Retrieves the horizontal resolution of the image the packed NFRecord is made from.
NFRecordGetImpressionType	Retrieves the impression type of the NFRecord.
NFRecordGetImpressionTypeMem	Retrieves the impression type of the packed NFRecord.
NFRecordGetMaxSize	Retrieves the maximal size of a packed NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.
NFRecordGetMaxSizeV1	Retrieves the maximal size of a packed in version 1.0 format NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.
NFRecordGetMinutia	Retrieves the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaCapacity	Retrieves the number of minutiae that the

	NFRecord can contain.
NFRecordGetMinutiaCount	Retrieves the number of minutiae in the NFRecord.
NFRecordGetMinutiaFormat	Retrieves the format of the minutiae in NFRecord.
NFRecordGetMinutiaNeighbour	Retrieves the minutia neighbour at the specified index of the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaNeighbourCount	Retrieves the number of minutia neighbours in the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaNeighbours	Copies all minutia neighbours of the minutia at the specified index of the NFRecord to the specified array.
NFRecordGetMinutiae	Copies all minutiae of NFRecord to the specified array.
NFRecordGetPatternClass	Retrieves the pattern class of the NFRecord.
NFRecordGetPatternClassMem	Retrieves the pattern class of the packed NFRecord.
NFRecordGetPosition	Retrieves the finger position of the NFRecord.
NFRecordGetPositionMem	Retrieves the finger position of the packed NFRecord.
NFRecordGetQuality	Retrieves the quality of the NFRecord.
NFRecordGetQualityMem	Retrieves the quality of the packed NFRecord.
NFRecordGetRidgeCountsType	Retrieves the ridge counts type the NFRecord contains.
NFRecordGetSize	Calculates packed size of the NFRecord.
NFRecordGetSizeV1	Calculates packed in version 1.0 format size of the NFRecord.
NFRecordGetVertResolution	Retrieves the vertical resolution of the image the NFRecord is made from.
NFRecordGetVertResolutionMem	Retrieves the vertical resolution of the image the packed NFRecord is made from.

Copyright © 2006 Neurotechnologija

NFRecordGetWidth	Retrieves the width of the image the NFRecord is made from.
NFRecordGetWidthMem	Retrieves the width of the image the packed NFRecord is made from.
NFRecordInfoDispose	For internal use.
NFRecordInsertCore	Inserts a NFCore into the NFRecord at the specified index.
NFRecordInsertDelta	Inserts a NFDelta into the NFRecord at the specified index.
NFRecordInsertDoubleCore	Inserts a NFDoubleCore into the NFRecord at the specified index.
NFRecordInsertMinutia	Inserts a NFMinutia into the NFRecord at the specified index.
NFRecordRemoveCore	Removes the core at the specified index of the NFRecord.
NFRecordRemoveDelta	Removes the delta at the specified index of the NFRecord.
NFRecordRemoveDoubleCore	Removes the double core at the specified index of the NFRecord.
NFRecordRemoveMinutia	Removes the minutia at the specified index of the NFRecord.
NFRecordSaveToMemory	Packs the NFRecord into the specified memory buffer.
NFRecordSaveToMemoryV1	Packs the NFRecord into the specified memory buffer in version 1.0 format.
NFRecordSetCore	Sets a NFCore at the specified index of the NFRecord.
NFRecordSetCoreCapacity	Sets the number of cores that the NFRecord can contain.
NFRecordSetDelta	Sets a NFDelta at the specified index of the NFRecord.
NFRecordSetDeltaCapacity	Sets the number of deltas that the NFRecord can contain.
NFRecordSetDoubleCore	Sets a NFDoubleCore at the specified index of the NFRecord.

Copyright © 2006 Neurotechnologija

NFRecordSetDoubleCoreCapacity	Sets the number of double cores that the NFRecord can contain.
NFRecordSetG	Sets the G of the NFRecord.
NFRecordSetImpressionType	Sets the impression type of the NFRecord.
NFRecordSetMinutia	Sets a NFMinutia at the specified index of the NFRecord.
NFRecordSetMinutiaCapacity	Sets the number of minutiae that the NFRecord can contain.
NFRecordSetMinutiaNeighbour	Sets a NFMinutiaNeighbour at the specified index of the minutia at the specified index of the NFRecord.
NFRecordSetMinutiaFormat	Sets the format of the minutiae in NFRecord.
NFRecordSetPatternClass	Sets the pattern class of the NFRecord.
NFRecordSetPosition	Sets the finger position of the NFRecord.
NFRecordSetQuality	Sets the quality of the NFRecord.
NFRecordSetRidgeCountsType	Sets the ridge counts type the NFRecord contains.

Structures

NFCore	Represents a core in a NFRecord.
NFDelta	Represents a delta in a NFRecord.
NFDoubleCore	Represents a double core in a NFRecord.
NFMinutia	Represents a minutia in a NFRecord.
NFMinutiaNeighbour	Represents a minutia neighbour in a NFRecord.
NFRecordInfo	For internal use.

Enumerations

NFImpressionType	Specifies the impression type.
NFMinutiaFormat	Specifies the minutia format.

Copyright © 2006 Neurotechnologija

NFMinutiaType	Specifies the minutia type.
NFPatternClass	Specifies the pattern class of the fingerprint.
NFPosition	Specifies the finger position.
NFRidgeCountsType	Specifies the type of ridge counts contained in a NFRecord.

Types

HNFRecord	Handle to NFRecord object.

Macros

NFR_BLOCK_SIZE	For internal use.
NFR_MAX_BLOCKED_ORIENTS_DIMENSION	For internal use.
NFR_MAX_CORE_COUNT	The maximum number of cores a NFRecord can contain.
NFR_MAX_DELTA_COUNT	The maximum number of deltas a NFRecord can contain.
NFR_MAX_DIMENSION	The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.
NFR_MAX_DOUBLE_CORE_COUNT	The maximum number of double cores a NFRecord can contain.
NFR_MAX_MINUTIA_COUNT	The maximum number of minutiae a NFRecord can contain.
NFR_NIST_RIDGE_COUNTS	For internal use.
NFR_RESOLUTION	The resolution of minutiae, cores, deltas and double cores coordinates in a NFRecord.
NFR_SAVE_BLOCKED_ORIENTS	The flag indicating whether blocked orientations should be packed in NFRecord.
NFR_SKIP_BLOCKED_ORIENTS	The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.

Copyright © 2006 Neurotechnologija

NFR_SKIP_CURVATURES	The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.
NFR_SKIP_GS	The flag indicating whether minutiae gs should be skipped while unpacking or packing NFRecord.
NFR_SKIP_QUALITIES	The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.
NFR_SKIP_RIDGE_COUNTS	The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.
NFR_SKIP_SINGULAR_POINTS	The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.

See Also

NFRecord Library

4.4.1.1. NFCore Structure

Represents a core in a NFRecord.

```
typedef struct NFCore_ { } NFCore;
```

Fields

Angle	Angle of this NFCore.
X	X coordinate of this NFCore.
Y	Y coordinate of this NFCore.

See Also

NFRecord Module

4.4.1.1.1. NFCore.Angle Field

Angle of this NFCore.

```
NInt Angle;
```

Remarks

The angle of the core is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the angle of the core is unknown.

See Also

NFCore

4.4.1.1.2. NFCore.X Field

X coordinate of this NFCore.

```
NUShort X;
```

Remarks

The x coordinate of the core is specified in pixels at NFR_RESOLUTION and *X* * [NFRecord horizontal resolution] / NFR_RESOLUTION can not be greater than NFR_MAX_DIMENSION or NFRecord width minus one.

See Also

NFCore

4.4.1.1.3. NFCore.Y Field

Y coordinate of this NFCore.

```
NUShort Y;
```

Remarks

The y coordinate of the core is specified in pixels at NFR_RESOLUTION and *Y* * [NFRecord vertical resolution] / NFR_RESOLUTION can not be greater than NFR_MAX_DIMENSION or NFRecord height minus one.

See Also

NFCore

4.4.1.2. NFDelta Structure

Represents a delta in a NFRecord.

Copyright © 2006 Neurotechnologija

```
typedef struct NFDelta_ { } NFDelta;
```

Fields

Angle1	First angle of this NFDelta.
Angle2	Second angle of this NFDelta.
Angle3	Third angle of this NFDelta.
X	X coordinate of this NFDelta.
Y	Y coordinate of this NFDelta.

See Also

NFRecord Module

4.4.1.2.1. NFDelta.Angle1 Field

First angle of this NFDelta.

```
NInt Angle1;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

See Also

NFDelta

4.4.1.2.2. NFDelta.Angle2 Field

Second angle of this NFDelta.

```
NInt Angle2;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the second angle of the delta is unknown.

Copyright © 2006 Neurotechnologija

See Also

NFDelta

4.4.1.2.3. NFDelta.Angle3 Field

Third angle of this NFDelta.

```
NInt Angle3;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the third angle of the delta is unknown.

See Also

NFDelta

4.4.1.2.4. NFDelta.X Field

X coordinate of this NFDelta.

```
NUShort X;
```

Remarks

The x coordinate of the delta is specified in pixels at NFR_RESOLUTION and *X* * [NFRecord horizontal resolution] / NFR_RESOLUTION can not be greater than NFR_MAX_DIMENSION or NFRecord width minus one.

See Also

NFDelta

4.4.1.2.5. NFDelta.Y Field

Y coordinate of this NFDelta.

```
NUShort Y;
```

Remarks

The y coordinate of the delta is specified in pixels at NFR_RESOLUTION and *Y* * [NFRecord vertical resolution] / NFR_RESOLUTION can not be greater than

NFR_MAX_DIMENSION or NFRecord height minus one.

See Also

NFDelta

4.4.1.3. NFDoubleCore Structure

Represents a double core in a NFRecord.

```
typedef struct NFDoubleCore_ { } NFDoubleCore;
```

Fields

X	X coordinate of this NFDoubleCore.
Y	Y coordinate of this NFDoubleCore.

See Also

NFRecord Module

4.4.1.3.1. NFDoubleCore.X Field

X coordinate of this NFDoubleCore.

```
NUShort X;
```

Remarks

The x coordinate of the double core is specified in pixels at NFR_RESOLUTION and *X* * [NFRecord horizontal resolution] / NFR_RESOLUTION can not be greater than NFR_MAX_DIMENSION or NFRecord width minus one.

See Also

NFDoubleCore

4.4.1.3.2. NFDoubleCore.Y Field

Y coordinate of this NFDoubleCore.

```
NUShort Y;
```

Remarks

The y coordinate of the double core is specified in pixels at NFR_RESOLUTION and *Y* * [NFRecord vertical resolution] / NFR_RESOLUTION can not be greater than NFR_MAX_DIMENSION or NFRecord height minus one.

See Also

NFDoubleCore

4.4.1.4. NFImpressionType Enumeration

Specifies the impression type.

```
typedef enum NFImpressionType_ { } NFImpressionType;
```

Members

`nfitLatentImpression`	Latent impression fingerprint.
`nfitLatentLift`	Latent lift fingerprint.
`nfitLatentPhoto`	Latent photo fingerprint.
`nfitLatentTracing`	Latent tracing fingerprint.
`nfitLiveScanContactless`	Live-scanned fingerprint using contactless device.
`nfitLiveScanPlain`	Live-scanned plain fingerprint.
`nfitLiveScanRolled`	Live-scanned rolled fingerprint.
`nfitNonliveScanPlain`	Nonlive-scanned (from paper) plain fingerprint.
`nfitNonliveScanRolled`	Nonlive-scanned (from paper) rolled fingerprint.
`nfitSwipe`	Live-scanned fingerprint by sliding the finger across a "swipe" sensor.

Remarks

This enumeration is implemented according to ANSI/NIST-ITL 1-2000 and ANSI INCITS 378-2004 standards.

See Also

NFRecord Module

4.4.1.5. NFMinutia Structure

Represents a minutia in a NFRecord.

```
typedef struct NFMinutia_ { } NFMinutia;
```

Fields

Angle	Angle of this NFMinutia.
Curvature	Ridge curvature near this NFMinutia.
G	G (ridge density) near this NFMinutia.
Quality	Quality of this NFMinutia.
Type	Type of this NFMinutia.
X	X coordinate of this NFMinutia.
Y	Y coordinate of this NFMinutia.

See Also

NFRecord Module

4.4.1.5.1. NFMinutia.Angle Field

Angle of this NFMinutia.

```
NByte Angle;
```

Remarks

The angle of the minutia is specified in 180/128 degrees units in counterclockwise order and can not be greater than 256 minus one.

See Also

NFMinutia

4.4.1.5.2. NFMinutia.Curvature Field

Ridge curvature near this NFMinutia.

```
NByte Curvature;
```

Remarks

If curvature of the minutia is unknown it must be set to 255.

See Also

NFMinutia

4.4.1.5.3. NFMinutia.G Field

G (ridge density) near this NFMinutia.

```
NByte G;
```

Remarks

If G of the minutia is unknown it must be set to 255.

See Also

NFMinutia

4.4.1.5.4. NFMinutia.Quality Field

Quality of this NFMinutia.

```
NByte Quality;
```

Remarks

The quality of the minutia must be in the range [0, 100]. The higher it is, the better the quality of the minutia is.

If quality of the minutia is unknown it must be set to zero.

See Also

NFMinutia

4.4.1.5.5. NFMinutia.Type Field

Type of this NFMinutia.

```
NFMinutiaType Type;
```

See Also

NFMinutia

4.4.1.5.6. NFMinutia.X Field

X coordinate of this NFMinutia.

```
NUShort X;
```

Remarks

The x coordinate of the minutia is specified in pixels at NFR_RESOLUTION and *X* * [NFRecord horizontal resolution] / NFR_RESOLUTION can not be greater than NFR_MAX_DIMENSION or NFRecord width minus one.

See Also

NFMinutia

4.4.1.5.7. NFMinutia.Y Field

Y coordinate of this NFMinutia.

```
NUShort Y;
```

Remarks

The y coordinate of the minutia is specified in pixels at NFR_RESOLUTION and *Y* * [NFRecord vertical resolution] / NFR_RESOLUTION can not be greater than NFR_MAX_DIMENSION or NFRecord height minus one.

See Also

NFMinutia

4.4.1.6. NFMinutiaFormat Enumeration

Specifies the minutia format.

This enumeration allows a bitwise combination of its member values.

```
typedef enum NFMinutiaFormat_ { } NFMinutiaFormat;
```

Members

`nfmfHasCurvature`	Indicates that NFMinutia. Curvature field contains meaningful value and is preserved during unpacking/packing of NFRecord.
`nfmfHasG`	Indicates that NFMinutia. G field contains

Copyright © 2006 Neurotechnologija

	meaningful value and is preserved during unpacking/packing of NFRecord.
`nfmfHasQuality`	Indicates that NFMinutia. Quality field contains meaningful value and is preserved during unpacking/packing of NFRecord.

See Also

NFRecord Module | NFMinutia

4.4.1.7. NFMinutiaNeighbour Structure

Represents a minutia neighbour in a NFRecord.

```
typedef struct NFMinutiaNeighbour_ { } NFMinutiaNeighbour;
```

Fields

`Index`	Index of neighbour minutia.
`RidgeCount`	Ridge count to neighbour minutia.

See Also

NFRecord Module

4.4.1.7.1. NFMinutiaNeighbour.Index Field

Index of neighbour minutia.

```
NInt Index;
```

See Also

NFMinutiaNeighbour

4.4.1.7.2. NFMinutiaNeighbour.RidgeCount Field

Ridge count to neighbour minutia.

```
NByte RidgeCount;
```

See Also

NFMinutiaNeighbour

4.4.1.8. NFMinutiaType Enumeration

Specifies the minutia type.

```
typedef enum NFMinutiaType_ { } NFMinutiaType;
```

Members

nfmtBifurcation	The minutia that is a bifurcation of a ridge.
nfmtEnd	The minutia that is an end of a ridge.
nfmtUnknown	The type of the minutia is unknown.

See Also

NFRecord Module | NFMinutia

4.4.1.9. NFPatternClass Enumeration

Specifies the pattern class of the fingerprint.

```
typedef enum NFPatternClass_ { } NFPatternClass;
```

Members

nfpcAccidentalWhorl	Accidental whorl pattern class.
nfpcAmputation	Amputation. Pattern class is not available.
nfpcCentralPocketLoop	Central pocket loop pattern class.
nfpcDoubleLoop	Double loop pattern class.
nfpcLeftSlantLoop	Left slant loop pattern class.
nfpcPlainArch	Plain arch pattern class.
nfpcPlainWhorl	Plain whorl pattern class.
nfpcRadialLoop	Radial loop pattern class.
nfpcRightSlantLoop	Right slant loop pattern class.
nfpcScar	Scar. Pattern class is not available.

Copyright © 2006 Neurotechnologija

nfpcTentedArch	Tented arch pattern class.
nfpcUlnarLoop	Ulnar loop pattern class.
nfpcUnknown	Unknown pattern class.
nfpcWhorl	Whorl pattern class.

Remarks

This enumeration is implemented according to ANSI/NIST-ITL 1-2000 standard.

See Also

NFRecord Module

4.4.1.10. NFPosition Enumeration

Specifies the finger position.

```
typedef enum NFPosition_ { } NFPosition;
```

Members

nfpLeftIndex	Index finger of the left hand.
nfpLeftLittle	Little finger of the left hand.
nfpLeftMiddle	Middle finger of the left hand.
nfpLeftRing	Ring finger of the left hand.
nfpLeftThumb	Thumb of the left hand.
nfpRightIndex	Index finger of the right hand.
nfpRightLittle	Little finger of the right hand.
nfpRightMiddle	Middle finger of the right hand.
nfpRightRing	Ring finger of the right hand.
nfpRightThumb	Thumb of the right hand.
nfpUnknown	Unknown finger.

Remarks

Copyright © 2006 Neurotechnologija

This enumeration is implemented according to ANSI/NIST-ITL 1-2000 and ANSI INCITS 378-2004 standards.

See Also

NFRecord Module

4.4.1.11. NFRecordAddCore Function

Adds a NFCore to the end of NFRecord cores.

```
NResult N_API NFRecordAddCore(
        HNFRecord hRecord,
        const NFCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[in] Pointer to the NFCore to add.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_INVALID_OPERATION	Number of cores in NFRecord (see NFRecordGetCoreCount) is equal to NFR_MAX_CORE_COUNT.

See Also

NFRecord Module | HNFRecord | NFCore

4.4.1.12. NFRecordAddDelta Function

Adds a NFDelta to the end of NFRecord deltas.

```
NResult N_API NFRecordAddDelta(
```

```
        HNFRecord hRecord,
        const NFDelta * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[in] Pointer to the NFDelta to add.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_INVALID_OPERATION	Number of deltas in NFRecord (see NFRecordGetDeltaCount) is equal to NFR_MAX_DELTA_COUNT.

See Also

NFRecord Module | HNFRecord | NFDelta

4.4.1.13. NFRecordAddDoubleCore Function

Adds a NFDoubleCore to the end of NFRecord double cores.

```
NResult N_API NFRecordAddDoubleCore(
        HNFRecord hRecord,
        const NFDoubleCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[in] Pointer to the NFDoubleCore to add.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_INVALID_OPERATION	Number of double cores in NFRecord (see NFRecordGetDoubleCoreCount) is equal to NFR_MAX_DOUBLE_CORE_COUNT.

See Also

NFRecord Module | HNFRecord | NFDoubleCore

4.4.1.14. NFRecordAddMinutia Function

Adds a NFMinutia to the end of NFRecord minutiae.

```
NResult N_API NFRecordAddMinutia(
        HNFRecord hRecord,
        const NFMinutia * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[in] Pointer to the NFMinutia to add.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Error Code	Condition
N_E_INVALID_OPERATION	Number of minutiae in NFRecord (see NFRecordGetMinutiaCount) is equal to NFR_MAX_MINUTIA_COUNT.

See Also

NFRecord Module | HNFRecord | NFMinutia

4.4.1.15. NFRecordCheck Function

Checks if format of the packed NFRecord is correct.

```
NResult N_API NFRecordCheck(
        const void * buffer,
        NSizeType bufferSize
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Copyright © 2006 Neurotechnologija

See Also

NFRecord Module

4.4.1.16. NFRecordClearCores Function

Removes all cores from the NFRecord.

```
NResult N_API NFRecordClearCores(
        HNFRecord hRecord
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord

4.4.1.17. NFRecordClearDeltas Function

Removes all deltas from the NFRecord.

```
NResult N_API NFRecordClearDeltas(
        HNFRecord hRecord
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord

4.4.1.18. NFRecordClearDoubleCores Function

Removes all double cores from the NFRecord.

```
NResult N_API NFRecordClearDoubleCores(
        HNFRecord hRecord
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord

4.4.1.19. NFRecordClearMinutiae Function

Removes all minutiae from the NFRecord.

```
NResult N_API NFRecordClearMinutiae(
        HNFRecord hRecord
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord

4.4.1.20. NFRecordClone Function

Creates a copy of the NFRecord.

```
NResult N_API NFRecordClone(
        HNFRecord hRecord,
        HNFRecord * pHClonedRecord
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pHClonedRecord	[out] Pointer to a HNFRecord that receives handle to newly created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pHClonedRecord* is NULL.

Error Code	Condition
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NFRecordFree` function.

See Also

NFRecord Module | HNFRecord | `NFRecordFree`

4.4.1.21. NFRecordCreate Function

Creates an empty NFRecord.

```
NResult N_API NFRecordCreate(
        NUShort width,
        NUShort height,
        NUShort horzResolution,
        NUShort vertResolution,
        NUInt flags,
        HNFRecord * pHRecord
);
```

Parameters

width	[in] Specifies width of fingerprint image.
height	[in] Specifies height of fingerprint image.
horzResolution	[in] Specifies horizontal resolution of fingerprint image.
vertResolution	[in] Specifies vertical resolution of fingerprint image.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

Copyright © 2006 Neurotechnologija

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

Error Code	Condition
N_E_ARGUMENT	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is zero.
N_E_ARGUMENT_NULL	*pHRecord* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- NFR_NIST_RIDGE_COUNTS

Created object must be deleted using NFRecordFree function.

See Also

NFRecord Module | HNFRecord | NFRecordFree

4.4.1.22. NFRecordCreateFromMemory Function

Unpacks a NFRecord from the specified memory buffer.

```
NResult N_API NFRecordCreateFromMemory(
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NFRecordInfo * pInfo,
        HNFRecord * pHRecord
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.

flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pInfo	[out] For internal use. Must be NULL.
pHRecord	[out] Pointer to HNFRecord that receives handle to newly created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHRecord* or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- NFR_NIST_RIDGE_COUNTS
- NFR_SKIP_BLOCKED_ORIENTS
- NFR_SKIP_CURVATURES
- NFR_SKIP_GS
- NFR_SKIP_QUALITIES
- NFR_SKIP_RIDGE_COUNTS
- NFR_SKIP_SINGULAR_POINTS

This function supports both NFRecord version 1.0 and 2.0 formats.

Created object must be deleted using `NFRecordFree` function.

See Also

NFRecord Module | HNFRecord | NFRecordInfo | `NFRecordFree` | `NFRecord-SaveToMemory`

Copyright © 2006 Neurotechnologija

4.4.1.23. NFRecordFree Function

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

```
void N_API NFRecordFree(
        HNFRecord hRecord
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.

Remarks

If *hRecord* is NULL, does nothing.

See Also

NFRecord Module | HNFRecord

4.4.1.24. NFRecordGetCore Function

Retrieves the core at the specified index of the NFRecord.

```
NResult N_API NFRecordGetCore(
        HNFRecord hRecord,
        NInt index,
        NFCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of core to retrieve.
pValue	[out] Pointer to NFCore that receives core.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount function.

See Also

NFRecord Module | HNFRecord | NFCore | NFRecordGetCoreCount | NFRecordSetCore

4.4.1.25. NFRecordGetCoreCapacity Function

Retrieves the number of cores that the NFRecord can contain.

```
NResult N_API NFRecordGetCoreCapacity(
        HNFRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NInt that receives number of cores NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Core capacity is the number of cores that the NFRecord can store. Core count (see NFRecordGetCoreCount function) is the number of cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding

cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | `NFRecordSetCoreCapacity` | `NFRecordGetCoreCount`

4.4.1.26. NFRecordGetCoreCount Function

Retrieves the number of cores in the NFRecord.

```
NResult N_API NFRecordGetCoreCount(
        HNFRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NInt that receives number of cores.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Core capacity (see `NFRecordGetCoreCapacity` and `NFRecordSetCoreCapacity` functions) is the number of cores that the NFRecord can store. Core count is the number of cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordGetCoreCapacity | NFRecordSetCoreCapacity

4.4.1.27. NFRecordGetCores Function

Copies all cores of NFRecord to the specified array.

```
NResult N_API NFRecordGetCores(
        HNFRecord hRecord,
        NFCore * arCores
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
arCores	[out] Pointer to array of NFCore that receives cores.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *arCores* is NULL.

Remarks

Array *arCores* points to must be large enough to receive all NFRecord cores. See NFRecordGetCoreCount function.

See Also

NFRecord Module | HNFRecord | NFCore | NFRecordGetCoreCount

4.4.1.28. NFRecordGetDelta Function

Retrieves the delta at the specified index of the NFRecord.

```
NResult N_API NFRecordGetDelta(
        HNFRecord hRecord,
        NInt index,
        NFDelta * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of delta to retrieve.
pValue	[out] Pointer to NFDelta that receives delta.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount function.

See Also

NFRecord Module | HNFRecord | NFDelta | NFRecordGetDeltaCount | NFRecordSetDelta

4.4.1.29. NFRecordGetDeltaCapacity Function

Retrieves the number of deltas that the NFRecord can contain.

```
NResult N_API NFRecordGetDeltaCapacity(
        HNFRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NInt that receives number of deltas NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Delta capacity is the number of deltas that the NFRecord can store. Delta count (see NFRecordGetDeltaCount function) is the number of deltas that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordSetDeltaCapacity | NFRecordGetDeltaCount

4.4.1.30. NFRecordGetDeltaCount Function

Retrieves the number of deltas in the NFRecord.

```
NResult N_API NFRecordGetDeltaCount(
        HNFRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NInt that receives number of deltas.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Delta capacity (see NFRecordGetDeltaCapacity and NFRecordSetDeltaCapacity functions) is the number of deltas that the NFRecord can store. Delta count is the number of deltas that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordGetDeltaCapacity | NFRecordSetDeltaCapacity

4.4.1.31. NFRecordGetDeltas Function

Copies all deltas of NFRecord to the specified array.

```
NResult N_API NFRecordGetDeltas(
        HNFRecord hRecord,
        NFDelta * arDeltas
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
arDeltas	[out] Pointer to array of NFDelta that receives deltas.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *arDeltas* is NULL.

Remarks

Array *arDeltas* points to must be large enough to receive all NFRecord deltas. See NFRecordGetDeltaCount function.

See Also

NFRecord Module | HNFRecord | NFDelta | NFRecordGetDeltaCount

4.4.1.32. NFRecordGetDoubleCore Function

Retrieves the double core at the specified index of the NFRecord.

```
NResult N_API NFRecordGetDoubleCore(
        HNFRecord hRecord,
        NInt index,
        NFDoubleCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of double core to retrieve.
pValue	[out] Pointer to NFDoubleCore that receives double core.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue*is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to double core count obtained using NFRecordGetDoubleCoreCount function.

See Also

NFRecord Module | HNFRecord | NFDoubleCore | NFRecordGetDoubleCoreCount |

NFRecordSetDoubleCore

4.4.1.33. NFRecordGetDoubleCoreCapacity Function

Retrieves the number of double cores that the NFRecord can contain.

```
NResult N_API NFRecordGetDoubleCoreCapacity(
        HNFRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NInt that receives number of double cores NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Double core capacity is the number of double cores that the NFRecord can store. Double core count (see NFRecordGetDoubleCoreCount function) is the number of double cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding double cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordSetDoubleCoreCapacity | NFRecord-GetDoubleCoreCount

4.4.1.34. NFRecordGetDoubleCoreCount Function

Retrieves the number of double cores in the NFRecord.

```
NResult N_API NFRecordGetDoubleCoreCount(
        HNFRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NInt that receives number of double cores.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue*is NULL.

Remarks

Double core capacity (see `NFRecordGetDoubleCoreCapacity` and `NFRecordSetDoubleCoreCapacity` functions) is the number of double cores that the NFRecord can store. Double core count is the number of double cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding double cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | `NFRecordGetDoubleCoreCapacity` | `NFRecordSetDoubleCoreCapacity`

4.4.1.35. NFRecordGetDoubleCores Function

Copies all double cores of NFRecord to the specified array.

```
NResult N_API NFRecordGetDoubleCores(
        HNFRecord hRecord,
        NFDoubleCore * arDoubleCores
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
arDoubleCores	[out] Pointer to array of NFDoubleCore that receives double cores.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *arDoubleCores* is NULL.

Remarks

Array *arDoubleCores* points to must be large enough to receive all NFRecord double cores. See `NFRecordGetDoubleCoreCount` function.

See Also

NFRecord Module | HNFRecord | NFDoubleCore | `NFRecordGetDoubleCoreCount`

4.4.1.36. NFRecordGetG Function

Retrieves the G of the NFRecord.

```
NResult N_API NFRecordGetG(
        HNFRecord hRecord,
        NByte * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NByte that receives G.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | NFRecordSetG

4.4.1.37. NFRecordGetGMem Function

Retrieves the G of the packed NFRecord.

```
NResult N_API NFRecordGetGMem(
        const void * buffer,
        NSizeType bufferSize,
        NByte * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NByte that receives G.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Copyright © 2006 Neurotechnologija

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

See Also

NFRecord Module

4.4.1.38. NFRecordGetHeight Function

Retrieves the height of the image the NFRecord is made from.

```
NResult N_API NFRecordGetHeight(
        HNFRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NUShort that receives height of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord

4.4.1.39. NFRecordGetHeightMem Function

Retrieves the height of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetHeightMem(
        const void * buffer,
        NSizeType bufferSize,
        NUShort * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NUShort that receives height of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `1` for version 1.0 format.

See Also

NFRecord Module

4.4.1.40. NFRecordGetHorzResolution Function

Retrieves the horizontal resolution of the image the NFRecord is made from.

```
NResult N_API NFRecordGetHorzResolution(
        HNFRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NUShort that receives horizontal resolution of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord

4.4.1.41. NFRecordGetHorzResolutionMem Function

Retrieves the horizontal resolution of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetHorzResolutionMem(
        const void * buffer,
        NSizeType bufferSize,
        NUShort * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NUShort that receives horizontal resolution of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `500` for version 1.0 format.

See Also

NFRecord Module

4.4.1.42. NFRecordGetImpressionType Function

Retrieves the impression type of the NFRecord.

```
NResult N_API NFRecordGetImpressionType(
        HNFRecord hRecord,
        NFImpressionType * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFImpressionType that receives impression type.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue*is NULL.

See Also

NFRecord Module | HNFRecord | NFImpressionType | `NFRecordSetImpressionType`

4.4.1.43. NFRecordGetImpressionTypeMem Function

Retrieves the impression type of the packed NFRecord.

```
NResult N_API NFRecordGetImpressionTypeMem(
        const void * buffer,
        NSizeType bufferSize,
        NFImpressionType * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NFImpressionType that receives impression type.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `nfitLiveScanPlain` for version 1.0 format.

See Also

NFRecord Module | NFImpressionType

4.4.1.44. NFRecordGetMaxSize Function

Retrieves the maximal size of a packed NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.

```
NResult N_API NFRecordGetMaxSize(
        NFMinutiaFormat minutiaFormat,
        NInt minutiaCount,
        NFRidgeCountsType ridgeCountsType,
        NInt coreCount,
        NInt deltaCount,
        NInt doubleCoreCount,
        NInt boWidth,
        NInt boHeight,
        NSizeType * pSize
);
```

Parameters

minutiaFormat	[in] The minutia format.
minutiaCount	[in] The number of minutiae.
ridgeCountsType	[in] The type of ridge counts.
coreCount	[in] The number of cores.
deltaCount	[in] The number of deltas.
doubleCoreCount	[in] The number of double cores.
boWidth	[in] The width of blocked orientations.
boHeight	[in] The height of blocked orientations.
pSize	[out] Pointer to NSizeType that receives maximal size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*minutiaFormat* is invalid.

Copyright © 2006 Neurotechnologija

Error Code	Condition
	- or - *ridgeCountsType* is invalid.
N_E_ARGUMENT_NULL	*pSize* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaCount* is less than zero or greater than or equal to NFR_MAX_MINUTIA_COUNT. - or - *coreCount* is less than zero or greater than or equal to NFR_MAX_CORE_COUNT. - or - *deltaCount* is less than zero or greater than or equal to NFR_MAX_DELTA_COUNT. - or - *doubleCoreCount* is less than zero or greater than or equal to NFR_MAX_DOUBLE_CORE_COUNT. - or - *boWidth* or *boHeight* is less than zero or greater than or equal to NFR_MAX_BLOCKED_ORIENTS_DIMENSION.

Remarks

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

The function calculates current (2.0) version packed size of NFRecord.

boWidth and *boHeight* parameters are for compatibility only. If one of them or both is zero, blocked orientations are ignored.

See Also

NFRecord Module | NFMinutiaFormat | NFMinutia | NFRidgeCountsType | NFCore | NF-

Delta | NFDoubleCore | NFRecordSaveToMemory

4.4.1.45. NFRecordGetMaxSizeV1 Function

Retrieves the maximal size of a packed in version 1.0 format NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.

```
NResult N_API NFRecordGetMaxSizeV1(
        NFMinutiaFormat minutiaFormat,
        NInt minutiaCount,
        NInt coreCount,
        NInt deltaCount,
        NInt doubleCoreCount,
        NInt boWidth,
        NInt boHeight,
        NSizeType * pSize
);
```

Parameters

minutiaFormat	[in] The minutia format.
minutiaCount	[in] The number of minutiae.
coreCount	[in] The number of cores.
deltaCount	[in] The number of deltas.
doubleCoreCount	[in] The number of double cores.
boWidth	[in] The width of blocked orientations.
boHeight	[in] The height of blocked orientations.
pSize	[out] Pointer to NSizeType that receives maximal size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*minutiaFormat* is invalid.
N_E_ARGUMENT_NULL	*pSize* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaCount* is less than zero or greater than or equal to

Error Code	Condition
	NFR_MAX_MINUTIA_COUNT. - or - *coreCount* is less than zero or greater than or equal to NFR_MAX_CORE_COUNT. - or - *deltaCount* is less than zero or greater than or equal to NFR_MAX_DELTA_COUNT. - or - *doubleCoreCount* is less than zero or greater than or equal to NFR_MAX_DOUBLE_CORE_COUNT. - or - *boWidth* or *boHeight* is less than zero or greater than or equal to NFR_MAX_BLOCKED_ORIENTS_DIMENSION.

Remarks

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

boWidth and *boHeight* parameters are for compatibility only. If one of them or both is zero, blocked orientations are ignored.

See Also

NFRecord Module | NFMinutiaFormat | NFMinutia | NFCore | NFDelta | NFDoubleCore | NFRecordSaveToMemoryV1

4.4.1.46. NFRecordGetMinutia Function

Retrieves the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordGetMinutia(
        HNFRecord hRecord,
        NInt index,
        NFMinutia * pValue
```

```
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of minutia to retrieve.
pValue	[out] Pointer to NFMinutia that receives minutia.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using `NFRecordGetMinutiaCount` function.

See Also

NFRecord Module | HNFRecord | NFMinutia | `NFRecordGetMinutiaCount` | `NFRecordGetMinutiae`

4.4.1.47. NFRecordGetMinutiaCapacity Function

Retrieves the number of minutiae that the NFRecord can contain.

```
NResult N_API NFRecordGetMinutiaCapacity(
        HNFRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NInt that receives number of minutiae NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Minutia capacity is the number of minutiae that the NFRecord can store. Minutia count (see `NFRecordGetMinutiaCount` function) is the number of minutiae that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | `NFRecordSetMinutiaCapacity` | `NFRecordGetMinutiaCount`

4.4.1.48. NFRecordGetMinutiaCount Function

Retrieves the number of minutiae in the NFRecord.

```
NResult N_API NFRecordGetMinutiaCount(
        HNFRecord hRecord,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NInt that receives number of minutiae.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Remarks

Minutia capacity (see `NFRecordGetMinutiaCapacity` and `NFRecordSetMinutiaCapacity` functions) is the number of minutiae that the NFRecord can store. Minutia count is the number of minutiae that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | `NFRecordGetMinutiaCapacity` | `NFRecordSetMinutiaCapacity`

4.4.1.49. NFRecordGetMinutiaFormat Function

Retrieves the format of the minutiae in NFRecord.

```
NResult N_API NFRecordGetMinutiaFormat(
        HNFRecord hRecord,
        NFMinutiaFormat * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFMinutiaFormat that receives format of minutiae in the NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | NFMinutiaFormat | `NFRecordSetMinutiaFormat`

4.4.1.50. NFRecordGetMinutiaNeighbour Function

Retrieves the minutia neighbour at the specified index of the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordGetMinutiaNeighbour(
        HNFRecord hRecord,
        NInt minutiaIndex,
        NInt index,
        NFMinutiaNeighbour * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
minutiaIndex	[in] The index of minutia.
index	[in] Index of minutia neighbour to retrieve.
pValue	[out] Pointer to NFMinutiaNeighbour that receives minutia neighbour.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaIndex* is less than zero or greater than or equal to minutia count obtained using `NFRecordGetMinutiaCount` function. - or - *index* is less than zero or greater than or equal to minutia neighbour count obtained using `NFRecordGetMinutiaNeighbourCount` function.

See Also

NFRecord Module | HNFRecord | NFMinutiaNeighbour | `NFRecordGetMinutiaCount` | `NFRecordGetMinutiaNeighbourCount` | `NFRecordSetMinutiaNeighbour`

4.4.1.51. NFRecordGetMinutiaNeighbourCount Function

Retrieves the number of minutia neighbours in the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordGetMinutiaNeighbourCount(
        HNFRecord hRecord,
        NInt minutiaIndex,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
minutiaIndex	[in] The index of minutia.
pValue	[out] Pointer to NInt that receives number of minutia neighbours.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using `NFRecordGetMinutiaCount` function.

See Also

NFRecord Module | HNFRecord

4.4.1.52. NFRecordGetMinutiaNeighbours Function

Copies all minutia neighbours of the minutia at the specified index of the NFRecord to the

specified array.

```
NResult N_API NFRecordGetMinutiaNeighbours(
        HNFRecord hRecord,
        NInt minutiaIndex,
        NFMinutiaNeighbour * arMinutiaNeighbours
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
minutiaIndex	[in] The index of minutia.
arMinutiaNeighbours	[out] Pointer to array of NFMinutiaNeighbour that receives minutia neighbours.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *arMinutiaNeighbours* is NULL.

Remarks

Array *arMinutiaNeighbours* points to must be large enough to receive all minutia neighbours. See `NFRecordGetMinutiaNeighbourCount` function.

See Also

NFRecord Module | HNFRecord | NFMinutiaNeighbour | `NFRecordGetMinutiaCount` | `NFRecordGetMinutiaNeighbourCount`

4.4.1.53. NFRecordGetMinutiae Function

Copies all minutiae of NFRecord to the specified array.

```
NResult N_API NFRecordGetMinutiae(
        HNFRecord hRecord,
        NFMinutia * arMinutiae
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
arMinutiae	[out] Pointer to array of NFMinutiathat receives minutiae.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *arMinutiae*is NULL.

Remarks

Array *arMinutiae* points to must be large enough to receive all NFRecord minutiae. See NFRecordGetMinutiaCount function.

See Also

NFRecord Module | HNFRecord | NFMinutia | NFRecordGetMinutiaCount

4.4.1.54. NFRecordGetPatternClass Function

Retrieves the pattern class of the NFRecord.

```
NResult N_API NFRecordGetPatternClass(
        HNFRecord hRecord,
        NFPatternClass * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFPatternClass that receives fingerprint pattern class.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | NFPatternClass | NFRecordSetPatternClass

4.4.1.55. NFRecordGetPatternClassMem Function

Retrieves the pattern class of the packed NFRecord.

```
NResult N_API NFRecordGetPatternClassMem(
        const void * buffer,
        NSizeType bufferSize,
        NFPatternClass * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NFPatternClass that receives fingerprint pattern class.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `nfpcUnknown` for version 1.0 format.

See Also

NFRecord Module | NFPatternClass

4.4.1.56. NFRecordGetPosition Function

Retrieves the finger position of the NFRecord.

```
NResult N_API NFRecordGetPosition(
        HNFRecord hRecord,
        NFPosition * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFPosition that receives finger position.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | NFPosition | `NFRecordSetPosition`

4.4.1.57. NFRecordGetPositionMem Function

Retrieves the finger position of the packed NFRecord.

```
NResult N_API NFRecordGetPositionMem(
        const void * buffer,
```

```
        NSizeType bufferSize,
        NFPosition * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NFPosition that receives finger position.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `nfpUnknown` for version 1.0 format.

See Also

NFRecord Module | NFPosition

4.4.1.58. NFRecordGetQuality Function

Retrieves the quality of the NFRecord.

```
NResult N_API NFRecordGetQuality(
        HNFRecord hRecord,
        NByte * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NByte that receives fingerprint quality.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | NFRecordSetQuality

4.4.1.59. NFRecordGetQualityMem Function

Retrieves the quality of the packed NFRecord.

```
NResult N_API NFRecordGetQualityMem(
        const void * buffer,
        NSizeType bufferSize,
        NByte * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NByte that receives fingerprint quality.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `0` for version 1.0 format.

See Also

NFRecord Module

4.4.1.60. NFRecordGetRidgeCountsType Function

Retrieves the ridge counts type the NFRecord contains.

```
NResult N_API NFRecordGetRidgeCountsType(
        HNFRecord hRecord,
        NFRidgeCountsType * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFRidgeCountsType that receives ridge counts type stored in NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue*is NULL.

See Also

NFRecord Module | HNFRecord | NFRidgeCountsType | NFRecordSetRidgeCountsType

4.4.1.61. NFRecordGetSize Function

Calculates packed size of the NFRecord.

```
NResult N_API NFRecordGetSize(
        HNFRecord hRecord,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL.

Remarks

The function calculates current (2.0) version packed size of NFRecord.

For the list of flags that are supported see NFRecordSaveToMemory function.

See Also

NFRecord Module | HNFRecord | NFRecordSaveToMemory

4.4.1.62. NFRecordGetSizeV1 Function

Copyright © 2006 Neurotechnologija

Calculates packed in version 1.0 format size of the NFRecord.

```
NResult N_API NFRecordGetSizeV1(
        HNFRecord hRecord,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL.

Remarks

For the list of flags that are supported see `NFRecordSaveToMemoryV1` function.

See Also

NFRecord Module | HNFRecord | `NFRecordSaveToMemoryV1`

4.4.1.63. NFRecordGetVertResolution Function

Retrieves the vertical resolution of the image the NFRecord is made from.

```
NResult N_API NFRecordGetVertResolution(
        HNFRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NUShort that receives vertical resolution of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord

4.4.1.64. NFRecordGetVertResolutionMem Function

Retrieves the vertical resolution of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetVertResolutionMem(
        const void * buffer,
        NSizeType bufferSize,
        NUShort * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NUShort that receives vertical resolution of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `500` for version 1.0 format.

See Also

NFRecord Module

4.4.1.65. NFRecordGetWidth Function

Retrieves the width of the image the NFRecord is made from.

```
NResult N_API NFRecordGetWidth(
        HNFRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NUShort that receives width of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord

4.4.1.66. NFRecordGetWidthMem Function

Retrieves the width of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetWidthMem(
        const void * buffer,
        NSizeType bufferSize,
        NUShort * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NUShort that receives width of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns 1 for version 1.0 format.

See Also

NFRecord Module

4.4.1.67. NFRecordInsertCore Function

Inserts a NFCore into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertCore(
        HNFRecord hRecord,
        NInt index,
        const NFCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index at which core is inserted.
pValue	[in] Pointer to the NFCore to insert.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than core count obtained using NFRecordGetCoreCount function.
N_E_INVALID_OPERATION	Number of cores in NFRecord (see NFRecordGetCoreCount) is equal to NFR_MAX_CORE_COUNT.

See Also

NFRecord Module | HNFRecord | NFCore | NFRecordGetCoreCount

4.4.1.68. NFRecordInsertDelta Function

Inserts a NFDelta into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertDelta(
        HNFRecord hRecord,
        NInt index,
        const NFDelta * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index at which delta is inserted.
pValue	[in] Pointer to the NFDelta to insert.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than delta count obtained using NFRecordGetDeltaCount function.
N_E_INVALID_OPERATION	Number of deltas in NFRecord (see NFRecordGetDeltaCount) is equal to NFR_MAX_DELTA_COUNT.

See Also

NFRecord Module | HNFRecord | NFDelta | NFRecordGetDeltaCount

4.4.1.69. NFRecordInsertDoubleCore Function

Inserts a NFDoubleCore into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertDoubleCore(
        HNFRecord hRecord,
        NInt index,
        const NFDoubleCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index at which double core is inserted.
pValue	[in] Pointer to the NFDoubleCore to insert.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than double core count obtained using NFRecordGetDoubleCoreCount function.
N_E_INVALID_OPERATION	Number of double core in NFRecord (see NFRecordGetDoubleCoreCount) is equal to NFR_MAX_DOUBLE_CORE_COUNT.

See Also

NFRecord Module | HNFRecord | NFDoubleCore | NFRecordGetDoubleCoreCount

4.4.1.70. NFRecordInsertMinutia Function

Inserts a NFMinutia into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertMinutia(
        HNFRecord hRecord,
        NInt index,
        const NFMinutia * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.

index	[in] Index at which minutia is inserted.
pValue	[in] Pointer to the NFMinutia to insert.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than minutia count obtained using NFRecord-GetMinutiaCount function.
N_E_INVALID_OPERATION	Number of minutia in NFRecord (see NFRecordGetMinutiaCount) is equal to NFR_MAX_MINUTIA_COUNT.

See Also

NFRecord Module | HNFRecord | NFMinutia | NFRecordGetMinutiaCount

4.4.1.71. NFRecordRemoveCore Function

Removes the core at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveCore(
        HNFRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of core to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount function.

See Also

NFRecord Module | HNFRecord | NFRecordGetCoreCount

4.4.1.72. NFRecordRemoveDelta Function

Removes the delta at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveDelta(
        HNFRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of delta to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount function.

See Also

NFRecord Module | HNFRecord | `NFRecordGetDeltaCount`

4.4.1.73. NFRecordRemoveDoubleCore Function

Removes the double core at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveDoubleCore(
        HNFRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of double core to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to double core count obtained using `NFRecordGetDoubleCoreCount` function.

See Also

NFRecord Module | HNFRecord | `NFRecordGetDoubleCoreCount`

4.4.1.74. NFRecordRemoveMinutia Function

Removes the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveMinutia(
        HNFRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of minutia to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

NFRecord Module | HNFRecord | NFRecordGetMinutiaCount

4.4.1.75. NFRecordSaveToMemory Function

Packs the NFRecord into the specified memory buffer.

```
NResult N_API NFRecordSaveToMemory(
        HNFRecord hRecord,
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
buffer	[out] Pointer to memory buffer to store packed NFRecord. Can be NULL.
bufferSize	[in] Size of memory buffer to store packed NFRecord.
flags	[in] Bitwise combination of zero or more

	flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store packed NFRecord.
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not zero.

Remarks

The function packs NFRecord in current (2.0) version format.

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as NFRecordGetSize function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with NFRecordGetSize function.

Note that blocked orientations are not packed by default.

The following flags are supported:

- NFR_SAVE_BLOCKED_ORIENTS
- NFR_SKIP_CURVATURES
- NFR_SKIP_GS
- NFR_SKIP_QUALITIES
- NFR_SKIP_RIDGE_COUNTS
- NFR_SKIP_SINGULAR_POINTS

See Also

NFRecord Module | HNFRecord | NFRecordGetSize | NFRecordCreateFrom-Memory

4.4.1.76. NFRecordSaveToMemoryV1 Function

Packs the NFRecord into the specified memory buffer in version 1.0 format.

```
NResult N_API NFRecordSaveToMemoryV1(
        HNFRecord hRecord,
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
buffer	[out] Pointer to memory buffer to store packed NFRecord. Can be NULL.
bufferSize	[in] Size of memory buffer to store packed NFRecord.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store packed NFRecord.
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not zero.

Copyright © 2006 Neurotechnologija

Remarks

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as NFRecordGetSizeV1 function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with NFRecordGetSizeV1 function.

Note that blocked orientations are not packed by default.

The following flags are supported:

- NFR_SAVE_BLOCKED_ORIENTS
- NFR_SKIP_CURVATURES
- NFR_SKIP_GS
- NFR_SKIP_SINGULAR_POINTS

See Also

NFRecord Module | HNFRecord | NFRecordCreateFromMemory | NFRecordGetSizeV1

4.4.1.77. NFRecordSetCore Function

Sets a NFCore at the specified index of the NFRecord.

```
NResult N_API NFRecordSetCore(
        HNFRecord hRecord,
        NInt index,
        const NFCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of core to set.
pValue	[in] Pointer to the NFCore to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount function.

See Also

NFRecord Module | HNFRecord | NFCore | NFRecordGetCoreCount | NFRecordGetCore

4.4.1.78. NFRecordSetCoreCapacity Function

Sets the number of cores that the NFRecord can contain.

```
NResult N_API NFRecordSetCoreCapacity(
        HNFRecord hRecord,
        NInt value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New number of cores NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than core count obtained using NFRecordGetCoreCount function.

Remarks

Core capacity is the number of cores that the NFRecord can store. Core count (see NFRecordGetCoreCount function) is the number of cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordGetCoreCapacity | NFRecordGetCoreCount

4.4.1.79. NFRecordSetDelta Function

Sets a NFDelta at the specified index of the NFRecord.

```
NResult N_API NFRecordSetDelta(
        HNFRecord hRecord,
        NInt index,
        const NFDelta * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of delta to set.
pValue	[in] Pointer to the NFDelta to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount function.

See Also

NFRecord Module | HNFRecord | NFDelta | NFRecordGetDeltaCount | NFRecord-GetDelta

4.4.1.80. NFRecordSetDeltaCapacity Function

Sets the number of deltas that the NFRecord can contain.

```
NResult N_API NFRecordSetDeltaCapacity(
        HNFRecord hRecord,
        NInt value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New number of deltas NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than delta count obtained using NFRecordGetDeltaCount function.

Remarks

Delta capacity is the number of deltas that the NFRecord can store. Delta count (see NFRecordGetDeltaCount function) is the number of deltas that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordGetDeltaCapacity | NFRecordGet-DeltaCount

4.4.1.81. NFRecordSetDoubleCore Function

Sets a NFDoubleCore at the specified index of the NFRecord.

```
NResult N_API NFRecordSetDoubleCore(
        HNFRecord hRecord,
        NInt index,
        const NFDoubleCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of double core to set.
pValue	[in] Pointer to the NFDoubleCore to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to double core count obtained using NFRecordGetDoubleCoreCount function.

See Also

NFRecord Module | HNFRecord | NFDoubleCore | NFRecordGetDoubleCoreCount | NFRecordGetDoubleCore

4.4.1.82. NFRecordSetDoubleCoreCapacity Function

Sets the number of double cores that the NFRecord can contain.

```
NResult N_API NFRecordSetDoubleCoreCapacity(
        HNFRecord hRecord,
        NInt value
```

```
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New number of double cores NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than double core count obtained using `NFRecordGetDoubleCoreCount` function.

Remarks

Double core capacity is the number of double cores that the NFRecord can store. Double core count (see `NFRecordGetDoubleCoreCount` function) is the number of double cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding double cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | `NFRecordGetDoubleCoreCapacity` | `NFRecordGetDoubleCoreCount`

4.4.1.83. NFRecordSetG Function

Sets the G of the NFRecord.

```
NResult N_API NFRecordSetG(
        HNFRecord hRecord,
        NByte value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New G value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFRecordGetG

4.4.1.84. NFRecordSetImpressionType Function

Sets the impression type of the NFRecord.

```
NResult N_API NFRecordSetImpressionType(
        HNFRecord hRecord,
        NFImpressionType value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New impression type value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFImpressionType | NFRecordGetImpressionType

4.4.1.85. NFRecordSetMinutia Function

Sets a NFMinutia at the specified index of the NFRecord.

```
NResult N_API NFRecordSetMinutia(
        HNFRecord hRecord,
        NInt index,
        const NFMinutia * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of minutia to set.
pValue	[in] Pointer to the NFMinutia to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

NFRecord Module | HNFRecord | NFMinutia | NFRecordGetMinutiaCount | NFRe-

cordGetMinutia

4.4.1.86. NFRecordSetMinutiaCapacity Function

Sets the number of minutiae that the NFRecord can contain.

```
NResult N_API NFRecordSetMinutiaCapacity(
        HNFRecord hRecord,
        NInt value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New number of minutiae NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than minutia count obtained using NFRecordGetMinutiaCount function.

Remarks

Minutia capacity is the number of minutiae that the NFRecord can store. Minutia count (see NFRecordGetMinutiaCount function) is the number of minutiae that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordGetMinutiaCapacity | NFRecordGet-MinutiaCount

Copyright © 2006 Neurotechnologija

4.4.1.87. NFRecordSetMinutiaFormat Function

Sets the format of the minutiae in NFRecord.

```
NResult N_API NFRecordSetMinutiaFormat(
        HNFRecord hRecord,
        NFMinutiaFormat value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New minutia format value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFMinutiaFormat | NFRecordGetMinutiaFormat

4.4.1.88. NFRecordSetMinutiaNeighbour Function

Sets a NFMinutiaNeighbour at the specified index of the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordSetMinutiaNeighbour(
        HNFRecord hRecord,
        NInt minutiaIndex,
        NInt index,
        const NFMinutiaNeighbour * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.

minutiaIndex	[in] The index of minutia.
index	[in] Index of minutia neighbour to set.
pValue	[in] Pointer to the NFMinutiaNeighbour to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaIndex* is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function. - or - *index* is less than zero or greater than or equal to minutia neighbour count obtained using NFRecordGetMinutiaNeighbourCount function.

See Also

NFRecord Module | HNFRecord | NFMinutiaNeighbour | NFRecordGetMinutiaCount | NFRecordGetMinutiaNeighbourCount | NFRecordGetMinutiaNeighbour

4.4.1.89. NFRecordSetPatternClass Function

Sets the pattern class of the NFRecord.

```
NResult N_API NFRecordSetPatternClass(
        HNFRecord hRecord,
        NFPatternClass value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New fingerprint pattern class value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFPatternClass | NFRecordGetPatternClass

4.4.1.90. NFRecordSetPosition Function

Sets the finger position of the NFRecord.

```
NResult N_API NFRecordSetPosition(
        HNFRecord hRecord,
        NFPosition value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New finger position value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFPosition | NFRecordGetPosition

4.4.1.91. NFRecordSetQuality Function

Sets the quality of the NFRecord.

```
NResult N_API NFRecordSetQuality(
        HNFRecord hRecord,
        NByte value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New fingerprint quality value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFRecordGetQuality

4.4.1.92. NFRecordSetRidgeCountsType Function

Sets the ridge counts type the NFRecord contains.

```
NResult N_API NFRecordSetRidgeCountsType(
        HNFRecord hRecord,
        NFRidgeCountsType value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New ridge counts type value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFRidgeCountsType | NFRecordGetRidgeCountsType

4.4.1.93. NFRidgeCountsType Enumeration

Specifies the type of ridge counts contained in a NFRecord.

```
typedef enum NFRidgeCountsType_ { } NFRidgeCountsType;
```

Members

nfrctEightNeighbours	The NFRecord contains ridge counts to closest minutia in each of the eight sectors of each minutia. First sector starts at minutia angle.
nfrctEightNeighboursWithIndexes	The NFRecord contains ridge counts to eight neighbours of each minutia.
nfrctFourNeighbours	The NFRecord contains ridge counts to closest minutia in each of the four sectors of each minutia. First sector starts at minutia angle.
nfrctFourNeighboursWithIndexes	The NFRecord contains ridge counts to four neighbours of each minutia.

`nfrctNone`	The NFRecord does not contain ridge counts.
`nfrctUnspecified`	For internal use.

See Also

NFRecord Module

4.5. NFTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates.

Import library (Windows): `NFTemplate.dll.lib`.

DLL (Windows): `NFTemplate.dll`.

Shared object (Linux): `libNFTemplate.so`.

Requirements (Windows):

- `NCore.dll`.
- `NFRecord.dll`.

Requirements (Linux):

- `libNCore.so`.
- `libNFRecord.so`.

Modules

NFTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).

4.5.1. NFTemplate Module

Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).

Header file: `NFTemplate.h`

Functions

Copyright © 2006 Neurotechnologija

NFTemplateAddRecord	Adds an empty finger record to the NFTemplate.
NFTemplateAddRecordCopy	Adds a copy of the finger record to the NFTemplate.
NFTemplateAddRecordFromMemory	Unpacks a finger record from the specified memory buffer and adds it to the NFTemplate.
NFTemplateCalculateSize	Calculates the size of a packed NFTemplate containing the specified number of finger records of specified size.
NFTemplateCheck	Checks if format of the packed NFTemplate is correct.
NFTemplateClearRecords	Removes all finger records from the NFTemplate.
NFTemplateClone	Creates a copy of the NFTemplate.
NFTemplateCreate	Creates an empty NFTemplate.
NFTemplateCreateFromMemory	Unpacks a NFTemplate from the specified memory buffer.
NFTemplateFree	Deletes the NFTemplate. After the object is deleted the specified handle is no longer valid.
NFTemplateGetRecord	Retrieves the finger record at the specified index of the NFTemplate.
NFTemplateGetRecordCapacity	Retrieves the number of finger records that the NFTemplate can contain..
NFTemplateGetRecordCount	Retrieves the number of finger records in the NFTemplate.
NFTemplateGetRecordCountMem	Retrieves the number of finger records in the packed NFTemplate.
NFTemplateGetSize	Calculates packed size of the NFTemplate.
NFTemplateInfoDispose	For internal use.
NFTemplatePack	Packs packed finger records as NFTemplate into the specified memory buffer.
NFTemplateRemoveRecord	Removes the finger record at the specified index of the NFTemplate.

Copyright © 2006 Neurotechnologija

NFTemplateSaveToMemory	Packs the NFTemplate into the specified memory buffer.
NFTemplateSetRecordCapacity	Sets the number of finger records that the NFTemplate can contain.
NFTemplateUnpack	Unpacks packed finger records from the packed NFTemplate.

Structures

NFTemplateInfo	For internal use.

Types

HNFTemplate	Handle to NFTemplate object.

Macros

NFT_MAX_RECORD_COUNT	The maximum number of finger records NFTemplate can contain.
NFT_PROCESS_FIRST_RECORD_ONLY	The flag indicating whether only the first finger record should be unpacked or packed while unpacking or packing NFTemplate.

See Also

NFTemplate Library

4.5.1.1. NFTemplateAddRecord Function

Adds an empty finger record to the NFTemplate.

```
NResult N_API NFTemplateAddRecord(
        HNFTemplate hTemplate,
        NUShort width,
        NUShort height,
        NUShort horzResolution,
        NUShort vertResolution,
        NUInt flags,
        HNFRecord * pHRecord
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
width	[in] Specifies width of fingerprint image.
height	[in] Specifies height of fingerprint image.
horzResolution	[in] Specifies horizontal resolution of fingerprint image.
vertResolution	[in] Specifies vertical resolution of fingerprint image.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is zero.
N_E_ARGUMENT_NULL	*hTemplate* or *pHRecord* is NULL.
N_E_INVALID_OPERATION	Number of finger records in NFTemplate (see NFTemplateGetRecordCount) is equal to NFT_MAX_RECORD_COUNT.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see NFRecordCreate function.

Copyright © 2006 Neurotechnologija

See Also

NFTemplate Module | HNFTemplate | HNFRecord | `NFTemplateGetRecordCount` | `NFRecordCreate`

4.5.1.2. NFTemplateAddRecordCopy Function

Adds a copy of the finger record to the NFTemplate.

```
NResult N_API NFTemplateAddRecordCopy(
        HNFTemplate hTemplate,
        HNFRecord hSrcRecord,
        HNFRecord * pHRecord
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
hSrcRecord	[in] Handle to the NFRecord object.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *hSrcRecord* or *pHRecord* is NULL.
N_E_INVALID_OPERATION	Number of finger records in NFTemplate (see `NFTemplateGetRecordCount`) is equal to NFT_MAX_RECORD_COUNT.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

NFTemplate Module | HNFTemplate | HNFRecord | `NFTemplateGetRecordCount`

4.5.1.3. NFTemplateAddRecordFromMemory Function

Copyright © 2006 Neurotechnologija

Unpacks a finger record from the specified memory buffer and adds it to the NFTemplate.

```
NResult N_API NFTemplateAddRecordFromMemory(
        HNFTemplate hTemplate,
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        HNFRecord * pHRecord
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHRecord*, or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Number of finger records in NFTemplate (see NFTemplateGetRecordCount) is equal to NFT_MAX_RECORD_COUNT.
N_E_OUT_OF_MEMORY	There was not enough memory.

Copyright © 2006 Neurotechnologija

Remarks

For the list of flags that are supported see `NFRecordCreateFromMemory` function.

See Also

NFTemplate Module | HNFTemplate | HNFRecord | `NFTemplateGetRecordCount` | `NFRecordCreateFromMemory`

4.5.1.4. NFTemplateCalculateSize Function

Calculates the size of a packed NFTemplate containing the specified number of finger records of specified size.

```
NResult N_API NFTemplateCalculateSize(
        NInt recordCount,
        NSizeType * arRecordSizes,
        NSizeType * pSize
);
```

Parameters

recordCount	[in] The number of finger records.
arRecordSizes	[in] Pointer to array of NSizeType that contains packed sizes of each finger record.
pSize	[out] Pointer to NSizeType that receives calculated size of packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Any entry of the array *arRecordSizes* points to is less than minimal finger record size.
N_E_ARGUMENT_NULL	*arRecordSizes* or *pSize* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*recordCount* is less than zero or greater than NFT_MAX_RECORD_COUNT.

Copyright © 2006 Neurotechnologija

Remarks

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

Use `NFRecordGetMaxSize` function to calculate packed size of an individual finger record.

See Also

NFTemplate Module | `NFRecordGetMaxSize` | `NFTemplateSaveToMemory`

4.5.1.5. NFTemplateCheck Function

Checks if format of the packed NFTemplate is correct.

```
NResult N_API NFTemplateCheck(
        const void * buffer,
        NSizeType bufferSize
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer*buffer* points to is inconsistent with NFTemplate format.

See Also

NFTemplate Module

Copyright © 2006 Neurotechnologija

4.5.1.6. NFTemplateClearRecords Function

Removes all finger records from the NFTemplate.

```
NResult N_API NFTemplateClearRecords(
        HNFTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.

See Also

NFTemplate Module | HNFTemplate

4.5.1.7. NFTemplateClone Function

Creates a copy of the NFTemplate.

```
NResult N_API NFTemplateClone(
        HNFTemplate hTemplate,
        HNFTemplate * pHClonedTemplate
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
pHClonedTemplate	[out] Pointer to HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHClonedTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NFTemplateFree` function.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateFree`

4.5.1.8. NFTemplateCreate Function

Creates an empty NFTemplate.

```
NResult N_API NFTemplateCreate(
        HNFTemplate * pHTemplate
);
```

Parameters

pHTemplate	[out] Pointer to HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NFTemplateFree` function.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateFree`

4.5.1.9. NFTemplateCreateFromMemory Function

Unpacks a NFTemplate from the specified memory buffer.

```
NResult N_API NFTemplateCreateFromMemory(
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NFTemplateInfo * pInfo,
        HNFTemplate * pHTemplate
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pInfo	[out] For internal use. Must be NULL.
pHTemplate	[out] Pointer to HNFTemplate that receives handle to newly created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHTemplate* or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.

Copyright © 2006 Neurotechnologija

Error Code	Condition
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- NFT_PROCESS_FIRST_RECORD_ONLY
- Flags supported by `NFRecordCreateFromMemory` function are applied to each finger record contained in the NFTemplate.

Created object must be deleted using `NFTemplateFree` function.

See Also

NFTemplate Module | HNFTemplate | NFTemplateInfo | `NFTemplateFree` | `NFRecordCreateFromMemory` | `NFTemplateSaveToMemory`

4.5.1.10. NFTemplateFree Function

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

```
void N_API NFTemplateFree(
        HNFTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.

Remarks

If *hTemplate* is NULL, does nothing.

See Also

NFTemplate Module | HNFTemplate

4.5.1.11. NFTemplateGetRecord Function

Retrieves the finger record at the specified index of the NFTemplate.

```
NResult N_API NFTemplateGetRecord(
        HNFTemplate hTemplate,
        NInt index,
        HNFRecord * pValue
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
index	[in] Index of finger record to retrieve.
pValue	[out] Pointer to HNFRecord that receives handle to NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to finger record count obtained using `NFTemplateGetRecordCount` function.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateGetRecordCount`

4.5.1.12. NFTemplateGetRecordCapacity Function

Retrieves the number of finger records that the NFTemplate can contain..

```
NResult N_API NFTemplateGetRecordCapacity(
        HNFTemplate hTemplate,
        NInt * pValue
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
pValue	[out] Pointer to NInt that receives number of finger records NFTemplate can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pValue* is NULL.

Remarks

Finger record capacity is the number of finger records that the NFTemplate can store. Finger record count (see `NFTemplateGetRecordCount` function) is the number of finger records that are actually in the NFTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateSetRecordCapacity` | `NFTemplateGetRecordCount`

4.5.1.13. NFTemplateGetRecordCount Function

Retrieves the number of finger records in the NFTemplate.

```
NResult N_API NFTemplateGetRecordCount(
        HNFTemplate hTemplate,
        NInt * pValue
);
```

Copyright © 2006 Neurotechnologija

Parameters

hTemplate	[in] Handle to NFTemplate object.
pValue	[out] Pointer to NInt that receives number of finger records.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pValue* is NULL.

Remarks

Finger record capacity (see NFTemplateGetRecordCapacity and NFTemplate-SetRecordCapacity functions) is the number of finger records that the NFTemplate can store. Finger record count is the number of finger records that are actually in the NFTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFTemplate Module | HNFTemplate | NFTemplateGetRecordCapacity | NFTemplateSetRecordCapacity

4.5.1.14. NFTemplateGetRecordCountMem Function

Retrieves the number of finger records in the packed NFTemplate.

```
NResult N_API NFTemplateGetRecordCountMem(
        const void * buffer,
        NSizeType bufferSize,
        NInt * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains

	packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
pValue	[out] Pointer to NInt that receives number of finger records.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.

See Also

NFTemplate Module

4.5.1.15. NFTemplateGetSize Function

Calculates packed size of the NFTemplate.

```
NResult N_API NFTemplateGetSize(
        HNFTemplate hTemplate,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hTemplate	[in] Handle to NFTemplate object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives calculated size of packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pSize* is NULL.

Remarks

For the list of flags that are supported see `NFTemplateSaveToMemory` function.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateSaveToMemory`

4.5.1.16. NFTemplatePack Function

Packs packed finger records as NFTemplate into the specified memory buffer.

```
NResult N_API NFTemplatePack(
        NInt recordCount,
        const void * * arRecords,
        NSizeType * arRecordSizes,
        void * buffer,
        NSizeType bufferSize,
        NSizeType * pSize
);
```

Parameters

recordCount	[in] The number of finger records.
arRecords	[in] Pointer to array of void * that contains packed finger records.
arRecordSizes	[in] Pointer to array of NSizeType that contains packed sizes of each finger record.
buffer	[out] Pointer to memory buffer to store packed NFTemplate.
bufferSize	[in] Size of memory buffer to store packed NFTemplate.
pSize	[out] Pointer to NSizeType that receives size of packed NFTemplate.

Copyright © 2006 Neurotechnologija

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*bufferSize* is less than size required to store packed NFTemplate. - or - Any entry of the array *arRecordSizes* points to is less than minimal finger record size. - or - Any entry of the array *arRecords* points to is NULL. - or - Any entry of the array *arRecordSizes* points to is not equal to size stored in memory buffer accroding entry of the array *arRecords* points to, points to.
N_E_ARGUMENT_NULL	*arRecords* or *buffer* or *pSize* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*recordCount* is less than zero or greater than NFT_MAX_RECORD_COUNT.
N_E_FORMAT	Data in memory buffer any entry of the array *arRecords* points to is inconsistent with NFRecord format.

Remarks

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

bufferSize must not be less than value calculated with `NFTemplateCalculateSize` function with the same parameter values.

See Also

NFTemplate Module | `NFTemplateCalculateSize` | `NFTemplateUnpack`

Copyright © 2006 Neurotechnologija

4.5.1.17. NFTemplateRemoveRecord Function

Removes the finger record at the specified index of the NFTemplate.

```
NResult N_API NFTemplateRemoveRecord(
        HNFTemplate hTemplate,
        NInt index
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
index	[in] Index of finger record to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater or equal than records count.

See Also

NFTemplate Module | HNFTemplate | NFTemplateGetRecordCount

4.5.1.18. NFTemplateSaveToMemory Function

Packs the NFTemplate into the specified memory buffer.

```
NResult N_API NFTemplateSaveToMemoryEx(
        HNFTemplate hTemplate,
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hTemplate	[in] Handle to NFTemplate object.
buffer	[out] Pointer to memory buffer to store packed NFTemplate. Can be NULL.
bufferSize	[in] Size of memory buffer to store packed NFTemplate.
flags	[in] Bitwise combination of zero or more flags that control behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store packed NFTemplate.
N_E_ARGUMENT_NULL	*hTemplate* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not zero.

Remarks

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as NFTemplateGetSize function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with NFTemplateGetSize function.

The following flags are supported:

- NFT_PROCESS_FIRST_RECORD_ONLY
- Flags supported by NFRecordSaveToMemory function are applied to each finger record contained in the NFTemplate.

Copyright © 2006 Neurotechnologija

See Also

NFTemplate Module | HNFTemplate | `NFTemplateGetSize` | `NFRecordSaveToMemory` | `NFTemplateCreateFromMemory`

4.5.1.19. NFTemplateSetRecordCapacity Function

Sets the number of finger records that the NFTemplate can contain.

```
NResult N_API NFTemplateSetRecordCapacity(
        HNFTemplate hTemplate,
        NInt value
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
value	[in] New number of finger records NFTemplate can contain.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than finger record count obtained using `NFTemplateGetRecordCount` function.

Remarks

Finger record capacity is the number of finger records that the NFTemplate can store. Finger record count (see `NFTemplateGetRecordCount` function) is the number of finger records that are actually in the NFTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFTemplate Module | HNFTemplate | NFTemplateGetRecordCapacity | NFTemplateGetRecordCount

4.5.1.20. NFTemplateUnpack Function

Unpacks packed finger records from the packed NFTemplate.

```
NResult N_API NFTemplateUnpack(
        const void * buffer,
        NSizeType bufferSize,
        NByte * pMajorVersion,
        NByte * pMinorVersion,
        NUInt * pSize,
        NByte * pHeaderSize,
        NInt * pRecordCount,
        const void * * arRecords,
        NSizeType * arRecordSizes
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
pMajorVersion	[out] Pointer to NByte that receives major version of the packed NFTemplate. Can be NULL.
pMinorVersion	[out] Pointer to NByte that receives minor version of the packed NFTemplate. Can be NULL.
pSize	[out] Pointer to NUInt that receives size of the packed NFTemplate. Can be NULL.
pHeaderSize	[out] Pointer to NByte that receives header size of the packed NFTemplate. Can be NULL.
pRecordCount	[out] Pointer to NInt that receives number of finger records contained in the packed NFTemplate. Can be NULL.
arRecords	[out] Pointer to array of void * that receives pointers to packed finger records contained in the packed NFTemplate. Can be NULL.
arRecordSizes	[out] Pointer to array of NSizeType that re-

Copyright © 2006 Neurotechnologija

	ceives sizes of packed finger records contained in the packed NFTemplate. Can be NULL.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.

Remarks

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

This function should be first called with *arRecords* and *arRecordSizes* parameters set to NULL to receive finger record count through *pRecordCount* parameter. Then arrays of finger records count length should be allocated and this function should be called with theese array passed in *arRecords* and *arRecordSizes* parameters.

See Also

NFTemplate Module | NFTemplatePack

4.6. NTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

Import library (Windows): NTemplate.dll.lib.

DLL (Windows): NTemplate.dll.

Shared object (Linux): libNTemplate.so.

Requirements (Windows):

- NCore.dll.
- NFRecord.dll.

Copyright © 2006 Neurotechnologija

- `NFTemplate.dll`.

Requirements (Linux):

- `libNCore.so`.
- `libNFRecord.so`.
- `libNFTemplate.so`.

Modules

NTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Templates (NTemplates).

4.6.1. NTemplate Module

Provides functionality for packing, unpacking and editing Neurotechnologija Templates (NTemplates).

Header file: `NTemplate.h`.

Functions

`NTemplateAddFingers`	Adds an empty fingers template to the NTemplate.
`NTemplateAddFingersCopy`	Adds a copy of the fingers template to the NFTemplate.
`NTemplateAddFingersFromMemory`	Unpacks a fingers template from the specified memory buffer and adds it to the NFTemplate.
`NTemplateCalculateSize`	Calculates the size of a packed NTemplate containing fingers template of the specified size.
`NTemplateCheck`	Checks if format of the packed NTemplate is correct.
`NTemplateClear`	Removes all templates from the NTemplate.
`NTemplateClone`	Creates a copy of the NTemplate.
`NTemplateCreate`	Creates an empty NTemplate.
`NTemplateCreateFromMemory`	Unpacks a NTemplate from the specified memory buffer.

Copyright © 2006 Neurotechnologija

NTemplateFree	Deletes the NTemplate. After the object is deleted the specified handle is no longer valid.
NTemplateGetFingers	Retrieves the fingers template of the NTemplate.
NTemplateGetSize	Calculates packed size of the NTemplate.
NTemplateInfoDispose	For internal use.
NTemplatePack	Packs packed fingers template as NTemplate into the specified memory buffer.
NTemplateRemoveFingers	Removes fingers template from the NTemplate.
NTemplateSaveToMemory	Packs the NFTemplate into the specified memory buffer.
NTemplateUnpack	Unpacks packed fingers template from the packed NTemplate.

Structures

NTemplateInfo	For internal use.

Types

HNTemplate	Handle to NTemplate object.

See Also

NTemplate Library

4.6.1.1. NTemplateAddFingers Function

Adds an empty fingers template to the NTemplate.

```
NResult N_API NTemplateAddFingers(
        HNTemplate hTemplate,
        HNFTemplate * pHFingers
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
pHFingers	[out] Pointer to a HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHFingers* is NULL.
N_E_INVALID_OPERATION	NTemplate already contains fingers template. See `NTemplateGetFingers` function.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

NTemplate Module | HNTemplate | HNFTemplate | `NTemplateGetFingers`

4.6.1.2. NTemplateAddFingersCopy Function

Adds a copy of the fingers template to the NFTemplate.

```
NResult N_API NTemplateAddFingersCopy(
        HNTemplate hTemplate,
        HNFTemplate hSrcFingers,
        HNFTemplate * pHFingers
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
hSrcFingers	[in] Handle to the NFTemplate object.
pHFingers	[out] Pointer to a HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *hSrcFingers* or *pHFingers* is NULL.
N_E_INVALID_OPERATION	NTemplate already contains fingers template. See NTemplateGetFingers function.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

NTemplate Module | HNTemplate | HNFTemplate | NTemplateGetFingers

4.6.1.3. NTemplateAddFingersFromMemory Function

Unpacks a fingers template from the specified memory buffer and adds it to the NFTemplate.

```
NResult N_API NTemplateAddFingersFromMemory(
        HNTemplate hTemplate,
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        HNFTemplate * pHFingers
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHFingers	[out] Pointer to HNFTemplate that receives handle to created NFTemplate object.

Copyright © 2006 Neurotechnologija

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHFingers*, or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.
N_E_INVALID_OPERATION	NTemplate already contains fingers template. See NTemplateGetFingers function.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see NFTemplateCreateFromMemory function.

See Also

NTemplate Module | HNTemplate | HNFTemplate | NTemplateGetFingers | NFTemplateCreateFromMemory

4.6.1.4. NTemplateCalculateSize Function

Calculates the size of a packed NTemplate containing fingers template of the specified size.

```
NResult N_API NTemplateCalculateSize(
        NSizeType fingersTemplateSize,
        NSizeType * pSize
);
```

Parameters

fingersTemplateSize	[in] The size of packed fingers template.
pSize	[out] Pointer to NSizeType that receives calculated size of packed NTemplate.

Copyright © 2006 Neurotechnologija

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*fingersTemplateSize* is not zero and is less than minimal size of packed fingers template.
N_E_ARGUMENT_NULL	*pSize* is NULL.

Remarks

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

Use NFTemplateCalculateSize function to calculate packed size of a fingers template.

See Also

NFTemplate Module | NFTemplateCalculateSize | NTemplateSaveToMemory

4.6.1.5. NTemplateCheck Function

Checks if format of the packed NTemplate is correct.

```
NResult N_API NTemplateCheck(
        const void * buffer,
        NSizeType bufferSize
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NTemplate.
bufferSize	[in] Size of memory buffer that contains packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory *buffer* buffer points to is inconsistent with NTemplate format.

See Also

NTemplate Module

4.6.1.6. NTemplateClear Function

Removes all templates from the NTemplate.

```
NResult N_API NTemplateClear(
        HNTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.

See Also

NTemplate Module | HNTemplate

4.6.1.7. NTemplateClone Function

Creates a copy of the NTemplate.

```
NResult N_API NTemplateClone(
        HNTemplate hTemplate,
        HNTemplate * pHClonedTemplate
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
pHClonedTemplate	[out] Pointer to HNTemplate that receives handle to created NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHClonedTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

NTemplate Module | HNTemplate | NTemplateFree

4.6.1.8. NTemplateCreate Function

Creates an empty NTemplate.

```
NResult N_API NTemplateCreate(
        HNTemplate * pHTemplate
);
```

Parameters

pHTemplate	[out] Pointer to HNTemplate that receives handle to created NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NTemplateFree` function.

See Also

NTemplate Module | HNTemplate | `NTemplateFree`

4.6.1.9. NTemplateCreateFromMemory Function

Unpacks a NTemplate from the specified memory buffer.

```
NResult N_API NTemplateCreateFromMemory(
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NTemplateInfo * pInfo,
        HNTemplate * pHTemplate
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NTemplate.
bufferSize	[in] Size of memory buffer that contains packed NTemplate.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pInfo	[out] For internal use. Must be NULL.
pHTemplate	[out] Pointer to HNTemplate that receives handle to newly created NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHTemplate* or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NTemplateFree` function.

The following flags are supported:

- Flags supported by `NFTemplateCreateFromMemory` function are applied to fingers template (if) contained in the NTemplate.

See Also

NTemplate Module | HNTemplate | NTemplateInfo | `NTemplateFree` | `NFTemplate-CreateFromMemory` | `NTemplateSaveToMemory`

4.6.1.10. NTemplateFree Function

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

```
void N_API NTemplateFree(
        HNTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.

Remarks

If *hTemplate* is NULL, does nothing.

See Also

NTemplate Module | HNTemplate

4.6.1.11. NTemplateGetFingers Function

Retrieves the fingers template of the NTemplate.

```
NResult N_API NTemplateGetFingers(
        HNTemplate hTemplate,
        HNFTemplate * pValue
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
pValue	[out] Points to HNFTemplate that receives handle to NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pValue* is NULL.

Remarks

If the NTemplate does not contain fingers template, the returned handle is NULL.

See Also

NTemplate Module | HNTemplate

4.6.1.12. NTemplateGetSize Function

Calculates packed size of the NTemplate.

```
NResult N_API NTemplateGetSize(
        HNTemplate hTemplate,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives calculated size of packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pSize* is NULL.

Remarks

For the list of flags that are supported see `NTemplateSaveToMemory` function.

See Also

NTemplate Module | HNTemplate | `NTemplateSaveToMemory`

4.6.1.13. NTemplatePack Function

Packs packed fingers template as NTemplate into the specified memory buffer.

```
NResult N_API NTemplatePack(
        const void * fingersTemplate,
        NSizeType fingersTemplateSize,
        void * buffer,
        NSizeType bufferSize,
        NSizeType * pSize
);
```

Parameters

fingersTemplate	[in] Pointer to memory buffer that contains packed fingers template.
fingersTemplateSize	[in] Size of memory buffer that contains packed fingers template.

buffer	[out] Pointer to memory buffer to store packed NTemplate.
bufferSize	[in] Size of memory buffer to store packed NTemplate.
pSize	[out] Pointer to NSizeType that receives size of packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*bufferSize* is less than size required to store packed NTemplate. - or - *fingersTemplateSize* is less than minimal fingers template size. - or - *fingersTemplateSize* is not equal to size stored in memory buffer *fingersTemplate* points to.
N_E_ARGUMENT_NULL	*fingersTemplate* is NULL and *fingersTemplateSize* is not equal to zero. - or - *buffer* or *pSize* is NULL.
N_E_FORMAT	Data in memory buffer *fingersTemplate* points to is inconsistent with NFTemplate format.

Remarks

bufferSize must not be less than value calculated with NTemplateCalculateSize function with the same parameter values.

See Also

NFTemplate Module | `NTemplateCalculateSize` | `NTemplateUnpack`

4.6.1.14. NTemplateRemoveFingers Function

Removes fingers template from the NTemplate.

```
NResult N_API NTemplateRemoveFingers(
        HNTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.

Remarks

If the NTemplate does not contain fingers template, does nothing.

See Also

NFTemplate Module | HNTemplate

4.6.1.15. NTemplateSaveToMemory Function

Packs the NFTemplate into the specified memory buffer.

```
NResult N_API NTemplateSaveToMemory(
        HNTemplate hTemplate,
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
buffer	[out] Pointer to memory buffer to store packed NTemplate. Can be NULL.
bufferSize	[in] Size of memory buffer to store packed NTemplate.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store packed NTemplate.
N_E_ARGUMENT_NULL	*hTemplate* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not zero.

Remarks

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as `NTemplateGetSize` function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with `NTemplateGetSize` function.

The following flags are supported:

- Flags supported by `NFTemplateSaveToMemory` function are applied to fingers template (if) contained in the NTemplate.

Copyright © 2006 Neurotechnologija

See Also

NTemplate Module | HNTemplate | NTemplateGetSize | NFTemplateSaveToMemory | NTemplateCreateFromMemory

4.6.1.16. NTemplateUnpack Function

Unpacks packed fingers template from the packed NTemplate.

```
NResult N_API NTemplateUnpack(
        const void * buffer,
        NSizeType bufferSize,
        NByte * pMajorVersion,
        NByte * pMinorVersion,
        NUInt * pSize,
        NByte * pHeaderSize,
        const void * * pFingersTemplate,
        NSizeType * pFingersTemplateSize
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NTemplate.
bufferSize	[in] Size of memory buffer that contains packed NTemplate.
pMajorVersion	[out] Pointer to NByte that receives major version of the packed NTemplate. Can be NULL.
pMinorVersion	[out] Pointer to NByte that receives minor version of the packed NTemplate. Can be NULL.
pSize	[out] Pointer to NUInt that receives size of the packed NTemplate. Can be NULL.
pHeaderSize	[out] Pointer to NByte that receives header size of the packed NTemplate. Can be NULL.
pFingersTemplate	[out] Pointer to void * that receives pointer to packed fingers template contained in the packed NTemplate. Can be NULL.
pFingersTemplateSize	[out] Pointer to NSizeType that receives size of packed fingers template contained in the packed NTemplate. Can be NULL.

Copyright © 2006 Neurotechnologija

Return Values

If the function succeeds, the return value 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.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory *buffer* buffer points to is inconsistent with NTemplate format.

See Also

NTemplate Module | NTemplatePack

Chapter 5. Reference (.NET)

This chapter contains reference of all libraries included in Template Management and Conversion Add-On for .NET developers.

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

Libraries

Neurotec	Provides classes that provide infrastructure for Neurotechnologija components.
Neurotec.Biometrics.ANTemplate	Provides functionality for converting ANSI/NIST-ITL 1-2000 standard (ANTemplate) files to and/or from Neurotechnologija Finger Records (NFRecords), Neurotechnologija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).
Neurotec.Biometrics.FMRecord	Provides methods and properties for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates (FMRecords) to and/or from Neurotechnologija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).
Neurotec.Biometrics.NFRecord	
Neurotec.Biometrics.NFTemplate	Provides methods for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).
Neurotec.Biometrics.NTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Templates (NTemplates).

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

Copyright © 2006 Neurotechnologija

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 Nu-

Copyright © 2006 Neurotechnologija

	merator.

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

Copyright © 2006 Neurotechnologija

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.Biometrics.ANTemplate Library

Provides functionality for converting ANSI/NIST-ITL 1-2000 standard files to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

DLL: `Neurotec.Biometrics.ANTemplate.dll`.

Namespaces

Neurotec.Biometrics	.NET wrapper encapsulates management functions and properties of ANSI/NIST ITL-1-2000 standard template.

5.2.1. Neurotec.Biometrics Namespace

.NET wrapper encapsulates management functions and properties of ANSI/NIST ITL-1-2000 standard template.

Classes

ANTemplate	The class was designed to manipulate template of ANSI/NIST ITL-1-2000 standard.

5.2.1.1. ANTemplate Class

Constructors

ANTemplate Constructor	Initializes a new instance of the `ANTemplate` class.

Methods

Dispose	Releases the resources used by `ANTemplate`.
Save	Writes ANTemplate object to file.
ToNFTemplate	Creates NFTemplate object from `ANTem-`

	`plate`.
ToNTemplate	Creates NTemplate object from `ANTemplate`.

Constants

DllName	Name of DLL containing unmanaged part of this class.
MaxDimension	For internal use.

5.2.1.1.1. ANTemplate Constructor

Initializes a new instance of the `ANTemplate` class.

5.2.1.1.1.1. ANTemplate(string fileName)

Initializes a new instance of the `ANTemplate` class from file.

```
public ANTemplate(
        string fileName
);
```

Parameters

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

5.2.1.1.1.2. ANTemplate(NFRecord nfRecord, string tot, string dai, string ori, string tcn)

Initializes a new instance of the `ANTemplate` class from NFRecord.

```
public ANTemplate(
        NFRecord nfRecord,
        string tot,
        string dai,
        string ori,
        string tcn
);
```

Parameters

`nfRecord`	The NFRecord object.

tot	Specifies Type Of Transaction.
dai	Specifies Destination Agency Identifier.
ori	Specifies Originating Agency Identifier.
tcn	Specifies Transaction Control Number.

5.2.1.1.1.3. ANTemplate(NFTemplate nfTemplate, string tot, string dai, string ori, string tcn)

Initializes a new instance of the `ANTemplate` class from NFTemplate.

```
public ANTemplate(
        NFTemplate nfTemplate,
        string tot,
        string dai,
        string ori,
        string tcn
);
```

Parameters

nfTemplate	The NFTemplate object.
tot	Specifies Type Of Transaction.
dai	Specifies Destination Agency Identifier.
ori	Specifies Originating Agency Identifier.
tcn	Specifies Transaction Control Number.

5.2.1.1.1.4. ANTemplate(NTemplate nTemplate, string tot, string dai, string ori, string tcn)

Initializes a new instance of the `ANTemplate` class from NTemplate.

```
public ANTemplate(
        NTemplate nTemplate,
        string tot,
        string dai,
        string ori,
        string tcn
);
```

Parameters

nTemplate	The NTemplate object.
tot	Specifies Type Of Transaction.
dai	Specifies Destination Agency Identifier.
ori	Specifies Originating Agency Identifier.
tcn	Specifies Transaction Control Number.

5.2.1.1.2. Dispose Method

Releases the resources used by `ANTemplate`.

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

5.2.1.1.3. Save Method

Writes ANTemplate object to file.

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

Parameters

fileName	A string that contains the name of the file.

5.2.1.1.4. ToNFTemplate Method

Creates NFTemplate object from `ANTemplate`.

```
public NFTemplate ToNFTemplate();
```

Return Values

The NFTemplate object.

See Also

NFTemplate

5.2.1.1.5. ToNTemplate Method

Creates NTemplate object from `ANTemplate`.

```
public NTemplate ToNTemplate();
```

Return Values

The NTemplate object.

See Also

NTemplate

5.3. Neurotec.Biometrics.FMRecord Library

Provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

DLL: `Neurotec.Biometrics.FMRecord.dll`.

Namespaces

Neurotec.Biometrics	Contains classes that provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

5.3.1. Neurotec.Biometrics Namespace

Contains classes that provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

Classes

Class	Description
FmrFingerView	Provides methods and properties for editing and converting ANSI INCITS 378-2004 standard finger views of templates (FmrFingerViews) to Neurotechnologija Finger Records (NFRecords).
FmrFingerView.CoreCollection	Represents the collection of FmrCore.
FmrFingerView.DeltaCollection	Represents the collection of FmrDelta.
FmrFingerView.MinutiaCollection	Represents the collection of FmrMinutia.
FmrFinger-	Represents the collection of NFMinu-

Class	Description
View.MinutiaEightNeighboursCollection	tiaNeighbour.
FmrFinger-View.MinutiaFourNeighboursCollection	Represents the collection of NFMinu-tiaNeighbour.
FMRecord	Provides methods and properties for load-ing, editing, saving and converting ANSI INCITS 378-2004 standard templates (FMRecords) to and/or from Neurotechnolo-gija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).
FMRecord.FingerViewCollection	Represents the collection of FmrFingerView.
FMRecordInfo	For internal use.

Structures

Structure	Description
FmrCore	Core structure of ANSI/INCIST 378 2004 standard template.
FmrDelta	Core structure of ANSI/INCIST 378 2004 standard template.
FmrMinutia	Core structure of ANSI/INCIST 378 2004 standard template.

5.3.1.1. FmrCore Structure

Core structure of ANSI/INCIST 378 2004 standard template.

Constructors

FmrCore Constructor	Initializes a new instance of the `FmrCore` structure.

Properties

Angle	Gets or sets angle of core.

RawAngle	Gets or sets raw angle of core.
X	Gets or sets x coordinate of core.
Y	Gets or sets y coordinate of core.

Methods

ToString	Returns a String that represents the current structure.

5.3.1.1.1. FmrCore Constructor

Initializes a new instance of the `FmrCore` structure.

5.3.1.1.1.1. FmrCore(ushort,ushort)

```
public FmrCore(
        ushort x,
        ushort y
);
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.

5.3.1.1.1.2. FmrCore(ushort,ushort,int)

```
public FmrCore(
        ushort x,
        ushort y,
        int angle
);
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.
angle	The angle of the core.

Copyright © 2006 Neurotechnologija

5.3.1.1.1.3. FmrCore(ushort,ushort,double)

```
public FmrCore(
        ushort x,
        ushort y,
        double angle
);
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.
angle	The angle of the core.

5.3.1.1.2. Angle Property

Angle of this FmrCore.

```
public double Angle {get;}
```

Property value

The angle of the core.

See Also

FmrCore

5.3.1.1.3. RawAngle Property

```
public int RawAngle {get;}
```

Property value

The row angle of the core.

See Also

FmrCore

5.3.1.1.4. X Property

```
public int X {get;}
```

Property value

X coordinate of the core.

See Also

FmrCore

5.3.1.1.5. Y Property

```
public int Y {get;}
```

Property value

X coordinate of the core.

See Also

FmrCore

5.3.1.1.6. ToString Method

Returns a String that represents the current structure.

```
public override string ToString();
```

Return Values

A String that represents the current structure.

5.3.1.2. FmrDelta Structure

Delta structure of ANSI/INCIST 378 2004 standard template.

Constructors

FmrDelta Constructor	Initializes a new instance of the `FmrDelta` structure.

Properties

Angle1	First angle of this FmrDelta.

Angle2	Second angle of this FmrDelta.
Angle3	Third angle of this FmrDelta.
RawAngle1	First raw angle of this FmrDelta.
RawAngle2	Second raw angle of this FmrDelta.
RawAngle3	Third raw angle of this FmrDelta.
X	X coordinate of this FmrDelta.
Y	Y coordinate of this FmrDelta.

Methods

ToString	Returns a String that represents the current structure.

5.3.1.2.1. FmrDelta Constructor

Initializes a new instance of the `FmrDelta` structure.

5.3.1.2.1.1. FmrDelta(ushort,ushort)

```
public FmrDelta(
        ushort x,
        ushort y
);
```

Parameters

x	X coordinate of the delta.
y	Y coordinate of the delta.

5.3.1.2.1.2. FmrDelta(ushort x, ushort y, int angle1, int angle2, int angle3)

```
public FmrDelta(
        ushort x,
        ushort y,
        int angle1,
        int angle2,
        int angle3
```

```
);
```

Parameters

x	X coordinate of the delta.
y	Y coordinate of the delta.
angle1	First angle of the delta.
angle2	Second angle of the delta.
angle3	Third angle of the delta.

5.3.1.2.1.3. public FmrDelta(ushort x, ushort y, double angle1, double angle2, double angle3)

```
public FmrDelta(
        ushort x,
        ushort y,
        double angle1,
        double angle2,
        double angle3
);
```

Parameters

x	X coordinate of the delta.
y	Y coordinate of the delta.
angle1	First angle of the delta.
angle2	Second angle of the delta.
angle3	Third angle of the delta.

5.3.1.2.2. Angle1 Property

First angle of this FmrDelta.

```
public double Angle1 {get;}
```

Property value

The first angle of the delta.

See Also

FmrDelta

5.3.1.2.3. Angle2 Property

Second angle of this FmrDelta.

```
public double Angle2 {get;}
```

Property value

The second angle of the delta.

See Also

FmrDelta

5.3.1.2.4. Angle3 Property

Third angle of this FmrDelta.

```
public double Angle3 {get;}
```

Property value

The third angle of the delta.

See Also

FmrDelta

5.3.1.2.5. RawAngle1 Property

First raw angle of this FmrDelta.

```
public int RawAngle1 {get;}
```

Property value

The first raw angle of the delta.

See Also

FmrDelta

5.3.1.2.6. RawAngle2 Property

Second raw angle of this FmrDelta.

```
public int RawAngle2 {get;}
```

Property value

The second raw angle of the delta.

See Also

FmrDelta

5.3.1.2.7. RawAngle3 Property

Third raw angle of this FmrDelta.

```
public int RawAngle3 {get;}
```

Property value

The third raw angle of the delta.

See Also

FmrDelta

5.3.1.2.8. X Property

X coordinate of this FmrDelta.

```
public int X {get;}
```

Property value

X coordinate of the delta.

See Also

FmrDelta

5.3.1.2.9. Y Property

Y coordinate of this FmrDelta.

```
public int Y {get;}
```

Property value

Y coordinate of the delta.

See Also

FmrDelta

5.3.1.2.10. ToString Method

Returns a String that represents the current structure.

```
public override string ToString();
```

Return Values

A String that represents the current structure.

5.3.1.3. FmrFingerView Class

Provides methods and properties for editing and converting ANSI INCITS 378-2004 standard finger views of templates (FmrFingerViews) to Neurotechnologija Finger Records (NFRecords).

Properties

Cores	Gets cores collection.
Deltas	Gets deltas collection.
FingerPosition	Gets or sets finger position.
FingerQuality	Gets or sets fingerprint quality.
HasEightNeighbourRidgeCounts	Gets or sets a value indicating whether FM-Record finger view has ridge counts to eight neighbours of each minutia.
HasFourNeighbourRidgeCounts	Gets or sets a value indicating whether FM-Record finger view has ridge counts to four neighbour of each minutia.
ImpressionType	Gets or sets the impression type of the finger record.
Minutiae	Gets minutiae collection.
MinutiaeEightNeighbours	Gets minutia eight neighbours collection.
MinutiaeFourNeighbours	Gets minutia four neighbours collection.

Copyright © 2006 Neurotechnologija

ViewNumber	Gets or sets view number to `FmrFinger-View` object.

Methods

GetMaxSize	Retrieves the maximal size of a FmrFinger-View in stored FMRecord containing the specified number of minutiae, cores and deltas and specified ridge counts.
ToNFRecord	Creates NFRecord object from FmrFinger-View.

Constants

FlagNistRidgeCounts	For internal use.
FlagProcessAllExtendedData	For internal use.
FlagSkipNeurotecFields	The flag indicating whether fields of Neuro-technologija should be skipped while load-ing or saving FmrFingerView.
FlagSkipRidgeCounts	The flag indicating whether ridge counts should be skipped while loading or saving FmrFingerView.
FlagSkipSingularPoints	The flag indicating whether singular points (cores and deltas) should be skipped while loading or saving FmrFingerView.
FlagUseNeurotecFields	The flag indicating whether fields of Neuro-technologija should be used while loading or saving FmrFingerView.
MaxCoreCount	The maximum number of cores a Fm-rFingerView can contain.
MaxDeltaCount	The maximum number of deltas a Fm-rFingerView can contain.
MaxDimension	The maximum value for x and y coordinates of a minutia, core or delta in a FmrFinger-View.
MaxMinutiaCount	The maximum number of minutiae a Fm-rFingerView can contain.

Copyright © 2006 Neurotechnologija

5.3.1.3.1. Cores Property

Gets cores collection.

```
public FmrFingerView.CoreCollection Cores {get;}
```

Property value

The FmrFingerView.CoreCollection collection.

See Also

FmrFingerView.CoreCollection

5.3.1.3.2. Deltas Property

Gets deltas collection.

```
public FmrFingerView.DeltaCollection Deltas {get;}
```

Property value

The FmrFingerView.DeltaCollection collection.

See Also

FmrFingerView.DeltaCollection

5.3.1.3.3. FingerPosition Property

Gets or sets finger position.

```
public NFPosition FingerPosition {get; set;}
```

Property value

One of the NFPosition values. By default is NFPosition.Unknown.

See Also

NFPosition

5.3.1.3.4. FingerQuality Property

Gets or sets fingerprint quality.

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

Property value

Fingerprint quality.

5.3.1.3.5. HasEightNeighbourRidgeCounts Property

Gets or sets a value indicating whether FMRecord finger view has ridge counts to eight neighbours of each minutia.

```
public bool HasEightNeighbourRidgeCounts {get; set;}
```

Property value

`true` if FMRecord finger view has ridge counts to eight neighbours of each minutia; otherwise, `false`.

5.3.1.3.6. HasFourNeighbourRidgeCounts Property

Gets or sets a value indicating whether FMRecord finger view has ridge counts to four neighbour of each minutia.

```
public bool HasFourNeighbourRidgeCounts {get; set;}
```

Property value

`true` if FMRecord finger view has ridge counts to four neighbours of each minutia; otherwise, `false`.

5.3.1.3.7. ImpressionType Property

Gets or sets the impression type of the finger record.

```
public NFImpressionType ImpressionType {get; set;}
```

Property value

One of the NFImpressionType values. The default is NFImpressionType.LiveScanPlain.

See Also

NFImpressionType

5.3.1.3.8. Minutiae Property

Gets minutiae collection.

```
public FmrFingerView.MinutiaCollection Minutiae {get;}
```

Copyright © 2006 Neurotechnologija

Property value

A FmrFingerView.MinutiaCollection that contains minutiae.

See Also

FmrFingerView.MinutiaCollection

5.3.1.3.9. MinutiaeEightNeighbours Property

Gets minutia eight neighbours collection.

```
public FmrFingerView.MinutiaEightNeighboursCollection MinutiaeEightNeighbours {get;}
```

Property value

A FmrFingerView.MinutiaEightNeighboursCollection that contains minutiae of eight neighbours.

See Also

FmrFingerView.MinutiaEightNeighboursCollection

5.3.1.3.10. MinutiaeFourNeighbours Property

Gets minutia four neighbours collection.

```
public FmrFingerView.MinutiaFourNeighboursCollection MinutiaeFourNeighbours {get;}
```

Property value

A FmrFingerView.MinutiaFourNeighboursCollection that contains minutiae of four neighbours.

See Also

FmrFingerView.MinutiaFourNeighboursCollection

5.3.1.3.11. ViewNumber Property

Gets or sets view number to `FmrFingerView` object.

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

Property value

A view number.

5.3.1.3.12. GetMaxSize Method

5.3.1.3.12.1. GetMaxSize(int,bool,bool,bool,int,bool,int)

Retrieves the maximal size of a FmrFingerView in stored FMRecord containing the specified number of minutiae, cores and deltas and specified ridge counts.

```
public static int GetMaxSize(
        int minutiaCount,
        bool hasFourNeighbourRidgeCounts,
        bool hasEightNeighbourRidgeCounts,
        bool coresHasAngles,
        int coreCount,
        bool deltasHasAngles,
        int deltaCount
);
```

Parameters

minutiaCount	The number of minutiae.
hasFourNeighbourRidgeCounts	Indicates whether FMRecord finger view has "four neighbour ridge counts".
hasEightNeighbourRidgeCounts	Indicates whether FMRecord finger view has "eight neighbour ridge counts".
coresHasAngles	`true` if cores has angles; otherwise, `false`.
coreCount	The number of cores.
deltasHasAngles	`true` if deltas has angles; otherwise, `false`.
deltaCount	The number of deltas.

Return Values

The maximal size of a FmrFingerView.

5.3.1.3.12.2. GetMaxSize(int,bool,bool,bool,int,bool,int,bool,bool,int,int)

For internal use.

```
public static int GetMaxSize(
        int minutiaCount,
        bool hasFourNeighbourRidgeCounts,
        bool hasEightNeighbourRidgeCounts,
        bool coresHasAngles,
```

```
        int coreCount,
        bool deltasHasAngles,
        int deltaCount,
        bool hasNeurotecCurvatures,
        bool hasNeurotecGs,
        int neurotecBOWidth,
        int neurotecBOHeight
);
```

5.3.1.3.13. ToNFRecord Methods

Creates NFRecord object from FmrFingerView.

5.3.1.3.13.1. NFRecord ToNFRecord()

Creates NFRecord object from FmrFingerView.

```
public NFRecord ToNFRecord();
```

Return Values

The `NFRecord` object.

See Also

NFRecord

5.3.1.3.13.2. NFRecord ToNFRecord(uint flags)

Creates NFRecord object from FmrFingerView. Behavior is controlled through flags.

```
public NFRecord ToNFRecord(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The `NFRecord` object.

See Also

NFRecord

5.3.1.4. FmrFingerView.CoreCollection Class

Represents the collection of FmrCore.

Properties

Capacity	Gets or sets the number of elements that the `FmrFingerView.CoreCollection` can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the `FmrFinger-View.CoreCollection`.
Clear	Removes all elements from the `Fm-rFingerView.CoreCollection`.
CopyTo	Copies the elements of the `FmrFinger-View.CoreCollection` to an array, starting at a particular array index.
GetEnumerator	Returns an enumerator that can be used to iterate through the CoreCollection.
Insert	Inserts an element into the `FmrFinger-View.CoreCollection` at the specified index.
RemoveAt	Removes the element at the specified index within the `FmrFinger-View.CoreCollection`.

5.3.1.4.1. Capacity Property

Gets or sets the number of elements that the `FmrFingerView.CoreCollection` can contain.

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

Property value

The number of elements that the `FmrFingerView.CoreCollection` can contain.

See Also

Count

5.3.1.4.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection

5.3.1.4.3. FmrFingerView.CoreCollection.Item Property

Gets or sets the member from collection by index.

```
public FmrCore this[
        int index
] {get; set;};
```

Parameters

index	The index of the FmrCore structure in the collection to get or set.

Property value

A FmrCore structure.

See Also

Count | FmrCore

5.3.1.4.4. Add Method

Adds an object to the `FmrFingerView.CoreCollection`.

```
public int Add(
        FmrCore value
);
```

Parameters

value	The FmrCore to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

5.3.1.4.5. Clear Method

Removes all elements from the `FmrFingerView.CoreCollection`.

```
public virtual void Clear()
```

See Also

RemoveAt

5.3.1.4.6. CopyTo Method

Copies the elements of the `FmrFingerView.CoreCollection` to an array, starting at a particular array index.

```
public void CopyTo(
        Array array,
        int index
);
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

5.3.1.4.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the CoreCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the core collection.

5.3.1.4.8. Insert Method

Inserts an element into the `FmrFingerView.CoreCollection` at the specified index.

```
public void Insert(
        int index,
        FmrCore value
);
```

Parameters

index	The zero-based index location where the FmrCore is inserted.
value	The FmrCore to add to the collection.

See Also

Add

5.3.1.4.9. RemoveAt Method

Removes the element at the specified index within the `FmrFinger-View.CoreCollection`.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the FmrCore to remove.

See Also

Clear

5.3.1.5. FmrFingerView.DeltaCollection Class

Represents the collection of FmrDelta.

Properties

Capacity	Gets or sets the number of elements that the FmrFingerView.DeltaCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the FmrFingerView.DeltaCollection
Clear	Removes all elements from the FmrFingerView.DeltaCollection
CopyTo	Copies the elements of the `FmrFingerView.DeltaCollection` to an array, starting at a particular array index.
GetEnumerator	Returns an enumerator that can be used to iterate through the `DeltaCollection`.
Insert	Inserts an element into the FmrFingerView.DeltaCollection at the specified index.
RemoveAt	Removes the element at the specified index within the FmrFingerView.DeltaCollection.

5.3.1.5.1. Capacity Property

Gets or sets the number of elements that the FmrFingerView.DeltaCollection can contain.

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

Property value

The number of elements that the `FmrFingerView.DeltaCollection` can contain.

See Also

Count

5.3.1.5.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection

5.3.1.5.3. FmrFingerView.DeltaCollection.Item Property

Gets or sets the member from collection by index.

```
public FmrDelta this[
        int index
] {get; set;}
```

Property value

A FmrDelta structure.

See Also

Count | FmrDelta

5.3.1.5.4. Add Method

Adds an object to the FmrFingerView.DeltaCollection

```
public int Add(
        FmrDelta value
);
```

Parameters

value	The FmrDelta to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

5.3.1.5.5. Clear Method

Removes all elements from the FmrFingerView.DeltaCollection

```
public virtual void Clear();
```

See Also

RemoveAt

5.3.1.5.6. CopyTo Method

Copies the elements of the `FmrFingerView.DeltaCollection` to an array, starting at a particular array index.

```
public void CopyTo(
        Array array,
        int index
);
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

5.3.1.5.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the `DeltaCollection`.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the checked index collection.

5.3.1.5.8. Insert Method

Inserts an element into the FmrFingerView.DeltaCollection at the specified index.

```
public void Insert(
        int index,
        FmrDelta value
);
```

Parameters

index	The zero-based index location where the FmrDelta is inserted.
value	The FmrDelta to add to the collection.

See Also

Add

5.3.1.5.9. RemoveAt Method

Removes the element at the specified index within the FmrFingerView.DeltaCollection.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the FmrDelta to remove.

See Also

Clear

5.3.1.6. FmrFingerView.MinutiaCollection Class

Represents the collection of FmrMinutia.

Properties

Capacity	Gets or sets the number of elements that the `FmrFingerView.CoreCollection` can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the FmrFinger-

	View.MinutiaCollection
Clear	Removes all elements from the FmrFinger-View.MinutiaCollection.
CopyTo	Copies the elements of the `FmrFinger-View.MinutiaCollection` to an array, starting at a particular array index.
GetEnumerator	Returns an enumerator that can be used to iterate through the FmrFinger-View.MinutiaCollection.
Insert	Inserts an element into the FmrFinger-View.MinutiaCollection at the specified index.
RemoveAt	Removes the element at the specified index within the FmrFinger-View.MinutiaCollection.

5.3.1.6.1. Capacity Property

Gets or sets the number of elements that the `FmrFingerView.CoreCollection` can contain.

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

Property value

The number of elements that the `FmrFingerView.MinutiaCollection` can contain.

See Also

Count

5.3.1.6.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection

5.3.1.6.3. FmrFingerView.MinutiaCollection.Item Property

Gets or sets the member from collection by index.

```
public FmrMinutia this[
        int index
] {get; set;}
```

Property value

A FmrMinutia structure.

See Also

FmrMinutia | Count

5.3.1.6.4. Add Method

Adds an object to the FmrFingerView.MinutiaCollection

```
public int Add(
        FmrMinutia value
);
```

Parameters

value	The FmrMinutia to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

5.3.1.6.5. Clear Method

Removes all elements from the FmrFingerView.MinutiaCollection.

```
public virtual void Clear();
```

See Also

RemoveAt

5.3.1.6.6. CopyTo Method

Copies the elements of the `FmrFingerView.MinutiaCollection` to an array, starting

at a particular array index.

```
public void CopyTo(
        Array array,
        int index
);
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

5.3.1.6.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the FmrFingerView.MinutiaCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the checked index collection.

5.3.1.6.8. Insert Method

Inserts an element into the FmrFingerView.MinutiaCollection at the specified index.

```
public void Insert(
        int index,
        FmrMinutia value
);
```

Parameters

index	The zero-based index location where the FmrMinutia is inserted.
value	The FmrMinutia to add to the collection.

See Also

Add

5.3.1.6.9. RemoveAt Method

Removes the element at the specified index within the FmrFingerView.MinutiaCollection.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the FmrMinutia to remove.

See Also

Clear

5.3.1.7. FmrFingerView.MinutiaEightNeighboursCollection Class

Represents the collection of NFMinutiaNeighbour.

Properties

Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

GetEnumerator	Returns an enumerator that can be used to iterate through the FmrFinger-View.MinutiaEightNeighboursCollection.

5.3.1.7.1. Count Property

Gets the number of items in the collection.

```
public virtual int Count {get;}
```

Property value

The number of items in the collection

5.3.1.7.2. FmrFingerView.MinutiaEightNeighboursCollection.Item Property

Gets or sets the member from collection by index.

5.3.1.7.2.1. NFMinutiaNeighbour[] this[int] {get;}

```
public const NFMinutiaNeighbour[] this[
        int minutiaIndex
] {get;}
```

Parameters

minutiaIndex	The index of the specified minutia.

Property value

An NFMinutiaNeighbour structures array of specified minutia.

See Also

Count

5.3.1.7.2.2. NFMinutiaNeighbour this[int, int] {get; set;}

```
public const NFMinutiaNeighbour this[
        int minutiaIndex,
        int index
] {get;  set;}
```

Parameters

minutiaIndex	The index of the specified minutia.
index	The index of the specified neighbour minutia to retrieve from the collection.

Property value

A NFMinutiaNeighbour structure of specified neighbour minutia.

See Also

Count

5.3.1.7.3. GetEnumerator Method

Returns an enumerator that can be used to iterate through the FmrFinger-View.MinutiaEightNeighboursCollection.

```
public virtual IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the minutia eight neighbours collection.

5.3.1.8. FmrFingerView.MinutiaFourNeighboursCollection Class

Represents the collection of NFMinutiaNeighbour.

Properties

Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

GetEnumerator	Returns an enumerator that can be used to iterate through the FmrFinger-View.MinutiaFourNeighboursCollection.

5.3.1.8.1. Count Property

Gets the number of items in the collection.

```
public virtual int Count {get;}
```

Property value

The number of items in the collection

5.3.1.8.2. FmrFingerView.MinutiaFourNeighboursCollection.Item Property

Gets or sets the member from collection by index.

5.3.1.8.2.1. NFMinutiaNeighbour[] this[int] {get;}

```
public const NFMinutiaNeighbour[] this[
        int minutiaIndex
] {get;}
```

Parameters

minutiaIndex	The index of the specified minutia.

Property value

An NFMinutiaNeighbour structures array of specified minutia.

See Also

Count

5.3.1.8.2.2. NFMinutiaNeighbour this[int, int] {get; set;}

```
public const NFMinutiaNeighbour this[
        int minutiaIndex,
        int index
] {get; set;}
```

Parameters

minutiaIndex	The index of the specified minutia.
index	The index of the specified neighbour minutia to retrieve from the collection.

Property value

A NFMinutiaNeighbour structure of specified neighbour minutia.

See Also

Count

5.3.1.8.3. GetEnumerator Method

Returns an enumerator that can be used to iterate through the FmrFinger-View.MinutiaFourNeighboursCollection.

```
public virtual IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the minutia four neighbours collection.

5.3.1.9. FmrMinutia Structure

Minutia structure of ANSI/INCIST 378 2004 standard template.

Constructors

FmrMinutia Constructor	Initializes a new instance of the `FmrMinutia` structure.

Properties

Angle	Gets or sets the angle of the minutia.
Quality	Gets or sets quality of the minutia.
RawAngle	Gets or sets the raw angle of the minutia.
Type	Gets or sets the type of the minutia.
X	Gets or sets x coordinate of the minutia.
Y	Gets or sets y coordinate of the minutia.

Methods

ToString	Returns a String that represents the current structure.

5.3.1.9.1. FmrMinutia Constructor

5.3.1.9.1.1. FmrMinutia(ushort,ushort,NFMinutiaType,byte)

```
public FmrMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        byte angle
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

5.3.1.9.1.2. FmrMinutia(ushort,ushort,NFMinutiaType,double)

```
public FmrMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        double angle
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

5.3.1.9.1.3. FmrMinutia(ushort,ushort,NFMinutiaType,byte,byte)

```
public FmrMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        byte angle,
        byte quality
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

Copyright © 2006 Neurotechnologija

quality	The quality of the minutia.

5.3.1.9.1.4. FmrMinutia(ushort,ushort,NFMinutiaType,double,byte)

```
public FmrMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        double angle,
        byte quality
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.
quality	The quality of the minutia.

5.3.1.9.2. Angle Property

Gets or sets the angle of the minutia.

```
public double Angle {get;}
```

Property value

The angle of the minutia.

See Also

FmrMinutia

5.3.1.9.3. Quality Property

Gets or sets quality of the minutia.

```
public byte Quality {get;}
```

Property value

The quality of the minutia.

See Also

FmrMinutia

5.3.1.9.4. RawAngle Property

Gets or sets the raw angle of the minutia.

```
public byte RawAngle {get;}
```

Property value

The raw angle of the minutia.

See Also

FmrMinutia

5.3.1.9.5. Type Property

Gets or sets the type of the minutia.

```
public NFMinutiaType Type {get;}
```

Property value

The type of the minutia.

See Also

FmrMinutia

5.3.1.9.6. X Property

Gets or sets x coordinate of the minutia.

```
public int X {get;}
```

Property value

The x coordinate of the minutia.

See Also

FmrMinutia

5.3.1.9.7. Y Property

Gets or sets y coordinate of the minutia.

```
public int Y {get;}
```

Property value

The y coordinate of the minutia.

See Also

FmrMinutia

5.3.1.9.8. ToString Method

Returns a String that represents the current structure.

```
public override string ToString();
```

Return Values

A String that represents the current structure.

5.3.1.10. FMRecord Class

Provides methods and properties for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates (FMRecords) to and/or from Neurotechnologija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).

Constructors

FMRecord Constructor	Initializes a new instance of the `FMRecord` class.

Properties

CaptureEquipmentCompliance	Gets or sets Capture Equipment Compliance from FMRecord object.
CaptureEquipmentId	Gets or sets Capture Equipment Id from FMRecord object.
CbeffProductId	Retrieves CBEFF Product Id from FMRecord object.
FingerViews	Retrieves FMRecord.FingerViewCollection

	collection.
ResolutionX	Retrieves horizontal resolution of scanned image from FMRecord.
ResolutionY	Retrieves vertical resolution of scanned image from FMRecord object.
SizeX	Retrieves horizontal size of scanned image from FMRecord object.
SizeY	Retrieves vertical size of scanned image from FMRecord object.

Methods

CalculateSize	For internal use.
Clone	Creates `FMRecord` object from another `FMRecord` object.
Dispose	Releases the resources used by FMRecord.
GetSize	Calculates packed size of `FMRecord` object.
Save	Packs FMRecord object to byte array.
ToNFTemplate	Creates NFTemplate object from FMRecord.
ToNTemplate	Creates NTemplate object from FMRecord.

Constants

DllName	Name of DLL containing unmanaged part of this class.
FlagProcessFirstFingerViewOnly	The flag indicating whether only the first finger view should be loaded or saved while loading or saving FMRecord.
MaxFingerViewCount	The maximum number of finger views FMRecord can contain.

5.3.1.10.1. FMRecord Constructor

Initializes a new instance of the `FMRecord` class.

5.3.1.10.1.1. FMRecord(ushort sizeX, ushort sizeY, ushort resolutionX, ushort resolutionY)

```
public FMRecord(
        ushort sizeX,
        ushort sizeY,
        ushort resolutionX,
        ushort resolutionY
);
```

Parameters

sizeX	Specifies horizontal size of fingerprint image.
sizeY	Specifies vertical size of fingerprint image.
resolutionX	Specifies horizontal resolution of fingerprint image.
resolutionY	Specifies vertical resolution of fingerprint image.

5.3.1.10.1.2. FMRecord(byte[] buffer)

Initializes a new instance of the `FMRecord` class from packed `FMRecord`.

```
public FMRecord(byte[] buffer);
```

Parameters

buffer	The packed `FMRecord`.

5.3.1.10.1.3. FMRecord(byte[] buffer, uint flags)

Initializes a new instance of the `FMRecord` class from packed `FMRecord`. Behavior is controlled through flags.

```
public FMRecord(byte[] buffer, uint flags);
```

Parameters

buffer	The packed `FMRecord`.

flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

5.3.1.10.1.4. FMRecord(byte[] buffer, out FMRecordInfo info)

Initializes a new instance of the `FMRecord` class from packed `FMRecord`.

```
public FMRecord(byte[] buffer, out FMRecordInfo info);
```

Parameters

buffer	The packed `FMRecord`.
info	The `FMRecordInfo` object.

5.3.1.10.1.5. FMRecord(byte[] buffer, uint flags, out FMRecordInfo info)

Initializes a new instance of the `FMRecord` class from packed `FMRecord`. Behavior is controlled through flags.

```
public FMRecord(
        byte[] buffer,
        uint flags,
        out FMRecordInfo info
);
```

Parameters

buffer	The packed `FMRecord`.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
info	The `FMRecordInfo` object.

5.3.1.10.1.6. FMRecord(NFRecord nfRecord)

Initializes a new instance of the `FMRecord` class from another `FMRecord` object.

```
public FMRecord(
        NFRecord nfRecord
);
```

Parameters

nfRecord	The NFRecord object.

5.3.1.10.1.7. FMRecord(NFRecord nfRecord, uint flags)

Initializes a new instance of the `FMRecord` class from another `FMRecord` object. Behavior is controlled through flags.

```
public FMRecord(
        NFRecord nfRecord,
        uint flags
);
```

Parameters

nfRecord	The NFRecord object.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

5.3.1.10.1.8. FMRecord(NFTemplate nfTemplate)

Initializes a new instance of the `FMRecord` class from the `NFTemplate` object.

```
public FMRecord(
        NFTemplate nfTemplate
);
```

Parameters

nfTemplate	The NFTemplate object.

5.3.1.10.1.9. FMRecord(NFTemplate nfTemplate, uint flags)

Initializes a new instance of the `FMRecord` class from the `NFTemplate` object. Behavior is controlled through flags.

```
public FMRecord(
        NFTemplate nfTemplate,
        uint flags
);
```

Parameters

nfTemplate	The NFTemplate object.

flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

5.3.1.10.2. CaptureEquipmentCompliance Property

Gets or sets Capture Equipment Compliance from FMRecord object.

```
byte CaptureEquipmentCompliance {get; set;}
```

Property value

The Capture Equipment Compliance.

5.3.1.10.3. CaptureEquipmentId Property

Gets or sets Capture Equipment Id from FMRecord object.

```
ushort CaptureEquipmentId {get; set;}
```

Property value

The Capture Equipment Id.

5.3.1.10.4. CbeffProductId Property

Retrieves CBEFF Product Id from FMRecord object.

```
uint CbeffProductId {get, set;}
```

Property value

The CBEFF.

5.3.1.10.5. FingerViews Property

Retrieves FMRecord.FingerViewCollection collection.

```
public FMRecord.FingerViewCollection FingerViews {get;}
```

Property value

The FMRecord.FingerViewCollection collection.

See Also

FMRecord.FingerViewCollection

Copyright © 2006 Neurotechnologija

5.3.1.10.6. ResolutionX Property

Retrieves horizontal resolution of scanned image from FMRecord.

```
ushort ResolutionX {get;}
```

Property value

The horizontal resolution of scanned image.

See Also

ResolutionY

5.3.1.10.7. ResolutionY Property

Retrieves vertical resolution of scanned image from FMRecord object.

```
ushort ResolutionY {get;}
```

Property value

The vertical resolution of scanned image.

See Also

ResolutionX

5.3.1.10.8. SizeX Property

Retrieves horizontal size of scanned image from FMRecord object.

```
ushort SizeX {get;}
```

Property value

The horizontal size of scanned image.

See Also

SizeY

5.3.1.10.9. SizeY Property

Retrieves vertical size of scanned image from FMRecord object.

```
ushort SizeY {get;}
```

Property value

The vertical size of scanned image.

See Also

SizeX

5.3.1.10.10. Clone Method

Creates `FMRecord` object from another `FMRecord` object.

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

Return Values

See Also

FMRecord Constructor

5.3.1.10.11. Dispose Method

Releases the resources used by FMRecord.

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

5.3.1.10.12. GetSize Method

Calculates packed size of `FMRecord` object.

5.3.1.10.12.1. GetSize()

```
public int GetSize();
```

Return Values

The packed size of `FMRecord` object.

5.3.1.10.12.2. GetSize(uint flags)

Calculates packed size of `FMRecord` object. Behavior is controlled through flags.

```
public int GetSize(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The packed size of `FMRecord` object.

5.3.1.10.13. Save Method

Packs FMRecord object to byte array.

5.3.1.10.13.1. Save()

Packs FMRecord object to byte array.

```
public byte[] Save();
```

Return Values

The byte array with FMRecord object.

5.3.1.10.13.2. Save(uint flags)

Packs FMRecord object to byte array. Behavior is controlled through flags.

```
public byte[] Save(
                uint flags
        );
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The byte array with FMRecord object.

5.3.1.10.14. ToNFTemplate Methods

Creates NFTemplate object from FMRecord.

5.3.1.10.14.1. ToNFTemplate()

Creates NFTemplate object from FMRecord.

```
public NFTemplate ToNFTemplate();
```

Return Values

The NFTemplate object.

See Also

NFTemplate

5.3.1.10.14.2. ToNFTemplate(uint flags)

Creates NFTemplate object from FMRecord. Behavior is controlled through flags.

```
public NFTemplate ToNFTemplate(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The NFTemplate object.

See Also

NFTemplate

5.3.1.10.15. NTemplate Method

Creates NTemplate object from FMRecord.

5.3.1.10.15.1. ToNTemplate()

Creates NTemplate object from FMRecord.

```
public NTemplate ToNTemplate();
```

Return Values

The NTemplate object.

See Also

NTemplate

5.3.1.10.15.2. ToNTemplate(uint)

Creates NTemplate object from FMRecord. Behavior is controlled through flags.

```
public NTemplate ToNTemplate(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The NTemplate object.

See Also

NTemplate

5.3.1.11. FMRecord.FingerViewCollection Class

Represents the collection of FmrFingerView.

Properties

Capacity	Gets or sets the number of elements that the `FMRecord.FingerViewCollection` can contain.
Item	Gets the member from collection by index.

Methods

Add	Adds an object to the `FMRecord.FingerViewCollection`.
AddCopy	Copies an object to the `FMRecord.FingerViewCollection`

5.3.1.11.1. Capacity Property

Gets or sets the number of elements that the `FMRecord.FingerViewCollection` can contain.

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

Property value

The number of elements that the FMRecord.FingerViewCollection can contain.

5.3.1.11.2. FMRecord.FingerViewCollection.Item Property

Gets the member from collection by index.

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

Property value

The FmrFingerView object.

See Also

FmrFingerView

5.3.1.11.3. Add Method

Adds an object to the `FMRecord.FingerViewCollection`.

```
public FmrFingerView Add();
```

Return Values

The FmrFingerView object.

See Also

FmrFingerView

5.3.1.11.4. AddCopy Method

Copies an object to the `FMRecord.FingerViewCollection`

```
public FmrFingerView AddCopy (
        FmrFingerView srcFingerView
);
```

Copyright © 2006 Neurotechnologija

Parameters

srcFingerView	The FmrFingerView object.

Return Values

The FmrFingerView object.

See Also

FmrFingerView

5.4. Neurotec.Biometrics.NFRecord Library

Provides functionality for packing, unpacking and editing Neurotechnologija Finger Records.

DLL: `Neurotec.Biometrics.NFRecord.dll`.

Namespaces

Neurotec.Biometrics	Provides classes for packing, unpacking and editing Neurotechnologija Finger Records.

5.4.1. Neurotec.Biometrics Namespace

Provides classes for packing, unpacking and editing Neurotechnologija Finger Records.

Classes

NFRecord	Contains functionality for packing, unpacking and editing NFRecord information.
NFRecord.CoreCollection	Represents the collection of NFCore.
NFRecord.DeltaCollection	Represents the collection of NFDelta.
NFRecord.DoubleCoreCollection	Represents the collection of NFDoubleCore.
NFRecord.MinutiaCollection	Represents the collection of NFMinutia.
NFRecord.MinutiaNeighboursCollection	Represents the collection of NFMinutiaNeighbour.

Structures

Copyright © 2006 Neurotechnologija

NFCore	The structure contains information of core.
NFDelta	The structure contains information of delta.
NFDoubleCore	The structure contains information of double core.
NFMinutia	The structure contains information of minutia.
NFMinutiaNeighbour	The structure contains information of minutia neighbour.

Enumerations

NFImpressionType	Specifies the impression types.
NFMinutiaFormat	Specifies formats of minutia.
NFMinutiaType	Specifies types of minutia.
NFPatternClass	Specifies pattern class of the fingerprint.
NFPosition	Specifies finger position.
NFRidgeCountsType	Specifies type of ridge counts contained in NFRecord.

5.4.1.1. NFRecord Class

Constructors

NFRecord Constructor	Initializes a new instance of the `NFRecord` class.

Properties

Cores	Gets cores collection.
Deltas	Gets deltas collection.
DoubleCores	Gets double cores collection.
G	Gets or sets additional fingerprint coefficient.

Copyright © 2006 Neurotechnologija

Handle	Gets handle to unmanaged `NFRecord` object.
Height	Gets the height of fingerprint image.
HorzResolution	Gets horizontal resolution of fingerprint image.
ImpressionType	Gets or sets the impression type of the finger record.
Minutiae	Gets minutiae collection.
MinutiaeNeighbours	Gets minutia neighbours from NFRecord.
MinutiaFormat	Gets or sets minutia format minutia format from NFRecord.
PatternClass	Gets or sets pattern class.
Position	Gets or sets finger position.
Quality	Gets or sets fingerprint quality.
RidgeCountsType	Gets or sets ridge counts type.
VertResolution	Gets vertical resolution of fingerprint image.
Width	Gets width of fingerprint image.

Methods

Check	Checks if format of packed `NFRecord` is correct.
Clone	Creates `NFRecord` object from another `NFRecord` object.
Dispose	Releases the resources used by NFRecord.
FromHandle	Creates `NFRecord` object from handle.
GetG	Retrieves G from packed `NFRecord`.
GetHeight	Retrieves height of fingerprint image from packed `NFRecord`.
GetHorzResolution	Retrieves horizontal resolution of fingerprint image from packed `NFRecord`.
GetImpressionType	Retrieves impression type from packed

Copyright © 2006 Neurotechnologija

	NFRecord.
GetMaxSize	Calculates the maximal `NFRecord` size.
GetMaxSizeV1	Calculates the maximal version 1.0 `NFRecord` size.
GetPatternClass	Retrieves fingerprint pattern class from packed `NFRecord`.
GetPosition	Retrieves finger position from packed `NFRecord`.
GetQuality	Retrieves fingerprint quality from packed `NFRecord`.
GetSize	Calculates packed size of `NFRecord`.
GetSizeV1	Calculates version 1.0 packed size of NFRecord.
GetVertResolution	Retrieves vertical resolution of fingerprint image from packed `NFRecord`.
GetWidth	Retrieves width of fingerprint image from packed `NFRecord`.
Save	Packs `NFRecord` to byte array.
SaveV1	Packs `NFRecord` to memory buffer in version 1.0 format.

Constants

DllName	Name of DLL containing unmanaged part of this class.
FlagNistRidgeCounts	For internal use.
FlagSaveBlockedOrients	The flag indicating whether blocked orientations should be packed in NFRecord.
FlagSkipBlockedOrients	The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.
FlagSkipCurvatures	The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.
FlagSkipGs	The flag indicating whether minutiae gs

Copyright © 2006 Neurotechnologija

	should be skipped while unpacking or packing NFRecord.
FlagSkipQualities	The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.
FlagSkipRidgeCounts	The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.
FlagSkipSingularPoints	The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.
MaxCoreCount	The maximum number of cores a NFRecord can contain.
MaxDeltaCount	The maximum number of deltas a NFRecord can contain.
MaxDimension	The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.
MaxDoubleCoreCount	The maximum number of double cores a NFRecord can contain.
MaxMinutiaCount	The maximum number of minutiae a NFRecord can contain.
Resolution	The resolution of minutiae, cores, deltas and double cores coordinates in a NFRecord.

5.4.1.1.1. NFRecord Constructors

Initializes a new instance of the `NFRecord` class.

5.4.1.1.1.1. NFRecord(byte[], uint,out NFRecordInfo)

```
public NFRecord(
        byte[] buffer,
        uint flags,
        out NFRecordInfo info
);
```

Parameters

buffer	The packed `NFRecord` object.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
info	The `NFTemplateInfo` object.

5.4.1.1.1.2. NFRecord(byte[],out NFRecordInfo)

```
public NFRecord(
        byte[] buffer,
        out NFRecordInfo info
);
```

Parameters

buffer	The packed `NFRecord` object.
info	The `NFTemplateInfo` object.

5.4.1.1.1.3. NFRecord(byte[] buffer)

Initializes a new instance of the `NFRecord` class.

```
public NFRecord(
        byte[] buffer
);
```

Parameters

buffer	The packed `NFRecord` object.

5.4.1.1.1.4. NFRecord(byte[] buffer, uint flags)

Initializes a new instance of the `NFRecord` class.

```
public NFRecord(
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	The packed `NFRecord` object.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

5.4.1.1.1.5. NFRecord(ushort,ushort,ushort,ushort)

```
public NFRecord(
        ushort width,
        ushort height,
        ushort horzResolution,
        ushort vertResolution
);
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.

5.4.1.1.1.6. NFRecord(ushort,ushort,ushort,ushort,uint)

```
public NFRecord(
        ushort width,
        ushort height,
        ushort horzResolution,
        ushort vertResolution,
        uint flags
);
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

5.4.1.1.2. Cores Property

Gets cores collection.

```
public NFRecord.CoreCollection Cores {get;}
```

Property value

A NFRecord.CoreCollection that contains cores.

See Also

NFRecord.CoreCollection

5.4.1.1.3. Deltas Property

Gets deltas collection.

```
public NFRecord.DeltaCollection Deltas {get;}
```

Property value

A NFRecord.DeltaCollection that contains deltas.

See Also

NFRecord.DeltaCollection

5.4.1.1.4. DoubleCores Property

```
public NFRecord.DoubleCoreCollection DoubleCores {get;}
```

Property value

A NFRecord.DoubleCoreCollection that contains double cores.

See Also

NFRecord.DoubleCoreCollection

5.4.1.1.5. G Property

Gets or sets additional fingerprint coefficient.

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

Property value

Fingerprint coefficient.

Remarks

G - average fingerprint ridge density.

5.4.1.1.6. Handle Property

Gets handle to unmanaged `NFRecord` object.

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

Property value

A handle to unmanaged NFRecord object.

5.4.1.1.7. Height Property

Gets the height of fingerprint image.

```
public ushort Height {get; }
```

Property value

Height of fingerprint image.

See Also

Width

5.4.1.1.8. HorzResolution Property

Gets horizontal resolution of fingerprint image.

```
public ushort HorzResolution {get; }
```

Property value

Horizontal resolution of fingerprint image.

See Also

VertResolution

5.4.1.1.9. ImpressionType Property

Gets or sets the impression type of the finger record.

```
public NFImpressionType ImpressionType {get; set;}
```

Property value

One of the NFImpressionType values. The default is NFImpressionType.LiveScanPlain.

See Also

NFImpressionType

5.4.1.1.10. Minutiae Property

Gets minutiae collection.

```
public NFRecord.MinutiaCollection Minutiae {get;}
```

Property value

A NFRecord.MinutiaCollection that contains minutiae.

See Also

NFRecord.MinutiaCollection

5.4.1.1.11. MinutiaeNeighbours Property

Gets minutia neighbours format from NFRecord.

```
public MinutiaNeighboursCollection MinutiaeNeighbours {get;}
```

Property value

A MinutiaNeighboursCollection collection.

See Also

MinutiaNeighboursCollection

5.4.1.1.12. MinutiaFormat Property

Gets or sets minutia format minutia format from NFRecord.

```
public NFMinutiaFormat MinutiaFormat {get; set;}
```

Property value

One of the NFMinutiaFormat values. The default is NFMinutiaFormat.HasCurvature.

See Also

NFMinutiaFormat

5.4.1.1.13. PatternClass Property

Gets or sets pattern class.

```
public NFPatternClass PatternClass {get; set;}
```

Property value

One of the NFPatternClass values. By default is NFPatternClass.Unknown.

See Also

NFPatternClass

5.4.1.1.14. Position Property

Gets or sets finger position.

```
public NFPosition Position {get; set;}
```

Property value

One of the NFPosition values. By default is NFPosition.Unknown.

See Also

NFPosition

5.4.1.1.15. Quality Property

Gets or sets fingerprint quality.

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

Property value

Fingerprint quality.

5.4.1.1.16. RidgeCountsType Property

Gets or sets ridge counts type.

Copyright © 2006 Neurotechnologija

```
public NFRidgeCountsType RidgeCountsType {get; set;}
```

Property value

One of the NFRidgeCountsType values.

See Also

NFRidgeCountsType

5.4.1.1.17. VertResolution Property

Gets vertical resolution of fingerprint image.

```
public ushort VertResolution {get;}
```

Property value

See Also

HorzResolution

5.4.1.1.18. Width Property

Gets width of fingerprint image.

```
public ushort Width {get;}
```

Property value

Width of fingerprint image.

See Also

Height

5.4.1.1.19. Check Methods

5.4.1.1.19.1. Check(byte[] buffer)

Checks if format of packed `NFRecord` is correct.

```
public static void Check(
        byte[] buffer
);
```

Parameters

buffer	The packed NFRecord object.

See Also

Check

5.4.1.1.19.2. Check(IntPtr buffer, int bufferSize)

Checks if format of packed NFRecord is correct.

```
public static void Check(
        IntPtr buffer,
        int bufferSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed NFRecord.
bufferSize	Size of memory buffer that contains packed NFRecord.

See Also

Check

5.4.1.1.20. Clone Method

Creates NFRecord object from another NFRecord object.

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

Return Values

The new NFRecord object.

5.4.1.1.21. Dispose Method

Releases the resources used by NFRecord.

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

5.4.1.1.22. FromHandle Method

Creates `NFRecord` object from handle.

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

Parameters

handle	Handle to unmanaged NFRecord object.

Return Values

`NFRecord` object.

See Also

NFRecord Constructors

5.4.1.1.23. GetG Method

Retrieves G from packed `NFRecord`.

```
public static byte GetG(
        byte[] buffer
);
```

Parameters

buffer	The packed `NFRecord` object.

Return Values

The G from packed `NFRecord`. G - average fingerprint ridge density.

See Also

G

5.4.1.1.24. GetHeight Method

Retrieves height of fingerprint image from packed NFRecord.

```
public static ushort GetHeight(
        byte[] buffer
);
```

Parameters

buffer	The packed `NFRecord` object.

Return Values

The height of fingerprint image.

See Also

GetWidth

5.4.1.1.25. GetHorzResolution Method

Retrieves horizontal resolution of fingerprint image from packed `NFRecord`.

```
public static ushort GetHorzResolution(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

The horizontal resolution of fingerprint image.

See Also

VertResolution

5.4.1.1.26. GetImpressionType Method

Retrieves impression type from packed NFRecord.

```
public static NFImpressionType GetImpressionType(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

A NFImpressionType enumeration member specifying impression type of fingerprint.

See Also

NFImpressionType | ImpressionType

5.4.1.1.27. GetMaxSize Method

Calculates the maximal `NFRecord` size.

5.4.1.1.27.1. GetMaxSize(NFMinutiaFormat,int,NFRidgeCountsType,int,int,int)

```
public static int GetMaxSize(
                NFMinutiaFormat minutiaFormat,
                int minutiaCount,
                NFRidgeCountsType ridgeCountsType,
                int coreCount,
                int deltaCount,
                int doubleCoreCount
        );
```

Parameters

minutiaFormat	One of the NFMinutiaFormat values.
minutiaCount	The minutiae count.
ridgeCountsType	One of the NFRidgeCountsType values.
coreCount	The cores count.
deltaCount	The deltas count.
doubleCoreCount	The double cores count.

Return Values

The maximal `NFRecord` size. The size depends on method parameters.

5.4.1.1.27.2. GetMaxSize(NFMinutiaFormat,int,NFRidgeCountsType,int,int,int,int,int)

```
public static int GetMaxSize(
                NFMinutiaFormat minutiaFormat,
                int minutiaCount,
                NFRidgeCountsType ridgeCountsType,
```

```
                int coreCount,
                int deltaCount,
                int doubleCoreCount,
                int boWidth,
                int boHeight
        );
```

Parameters

minutiaFormat	One of the NFMinutiaFormat values.
minutiaCount	The minutiae count.
ridgeCountsType	One of the NFRidgeCountsType values.
coreCount	The cores count.
deltaCount	The deltas count.
doubleCoreCount	The double cores count.
boWidth	For compatibility with VeriFinger.
boHeight	For compatibility with VeriFinger.

Return Values

The maximal `NFRecord` size. The size depends on method parameters.

5.4.1.1.28. GetMaxSizeV1 Method

Calculates the maximal version 1.0 `NFRecord` size.

5.4.1.1.28.1. GetMaxSizeV1(NFMinutiaFormat,int,int,int,int)

```
public static int GetMaxSizeV1(
        NFMinutiaFormat minutiaFormat,
        int minutiaCount,
        int coreCount,
        int deltaCount,
        int doubleCoreCount
);
```

Parameters

minutiaFormat	One of the NFMinutiaFormat values.
minutiaCount	The minutiae count.

coreCount	The cores count.
deltaCount	The deltas count.
doubleCoreCount	The double cores count.

Return Values

The maximal version 1.0 `NFRecord` size. The size depends on method parameters.

5.4.1.1.28.2. GetMaxSizeV1(NFMinutiaFormat,int,int,int,int,int,int)

```
public static int GetMaxSizeV1(
        NFMinutiaFormat minutiaFormat,
        int minutiaCount,
        int coreCount,
        int deltaCount,
        int doubleCoreCount,
        int boWidth,
        int boHeight
);
```

Parameters

minutiaFormat	One of the NFMinutiaFormat values.
minutiaCount	The minutiae count.
coreCount	The cores count.
deltaCount	The deltas count.
doubleCoreCount	The double cores count.
boWidth	For compatibility with VeriFinger.
boHeight	For compatibility with VeriFinger.

Return Values

The maximal version 1.0 `NFRecord` size. The size depends on method parameters.

5.4.1.1.29. GetPatternClass Method

Retrieves fingerprint pattern class from packed `NFRecord`.

```
public static NFPatternClass GetPatternClass(
        byte[] buffer
```

```
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

One of the NFPatternClass values.

See Also

PatternClass

5.4.1.1.30. GetPosition Method

Retrieves finger position from packed `NFRecord`.

```
public static NFPosition GetPosition(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

One of the NFPosition values.

See Also

NFPosition | Position

5.4.1.1.31. GetQuality Method

Retrieves fingerprint quality from packed `NFRecord`.

```
public static byte GetQuality(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

The value of fingerprint quality.

5.4.1.1.32. GetSize Methods

Calculates packed size of `NFRecord`.

5.4.1.1.32.1. GetSize()

Calculates packed size of NFRecord.

```
public int GetSize();
```

Return Values

The packed size of NFRecord.

Remarks

The method calculates current (2.0) version packed size of NFRecord.

See Also

NFRecord Constructors | `GetSize(uint flags)`

5.4.1.1.32.2. GetSize(uint flags)

Calculates packed size of NFRecord. Behavior is controlled through flags.

```
public int GetSize(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The packed size of NFRecord.

Remarks

The method calculates current (2.0) version packed size of NFRecord.

See Also

NFRecord Constructors | `GetSize()`

5.4.1.1.33. GetSizeV1 Method

Calculates version 1.0 packed size of NFRecord.

5.4.1.1.33.1. GetSizeV1()

Calculates version 1.0 packed size of NFRecord.

```
public int GetSizeV1()
```

Return Values

The packed size of NFRecord.

5.4.1.1.33.2. GetSizeV1(uint)

Calculates version 1.0 packed size of NFRecord. Behavior is controlled through flags.

```
public int GetSizeV1(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The packed size of NFRecord.

See Also

NFRecord Constructors | `GetSize`

5.4.1.1.34. GetVertResolution Method

Retrieves vertical resolution of fingerprint image from packed `NFRecord`.

```
public static ushort GetVertResolution(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

The vertical resolution of fingerprint image.

See Also

GetHorzResolution

5.4.1.1.35. GetWidth Method

Retrieves width of fingerprint image from packed `NFRecord`.

```
public static ushort GetWidth(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

The width of fingerprint image.

See Also

GetHeight

5.4.1.1.36. Save Methods

Packs `NFRecord` to byte array.

5.4.1.1.36.1. Save()

Packs NFRecord to byte array.

```
public byte[] Save();
```

Return Values

The array of packed NFRecord.

Remarks

The method packs NFRecord in current (2.0) version format.

Note that blocked orientations are not packed. To pack them use Save with flags or SaveV1 method.

See Also

NFRecord Constructors

5.4.1.1.36.2. Save(uint flags)

Packs NFRecord to byte array. Behavior is controlled through flags.

```
public byte[] Save(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The byte array of packed NFRecord.

Remarks

The method packs NFRecord in current (2.0) version format.

Note that blocked orientations are not packed by default.

The following flags are supported:

See Also

NFRecord Constructors | Save | SaveV1

5.4.1.1.37. SaveV1 Method

Packs `NFRecord` to memory buffer in version 1.0 format.

5.4.1.1.37.1. SaveV1()

```
public byte[] SaveV1();
```

Return Values

The byte array of packed NFRecord.

See Also

Save

5.4.1.1.37.2. SaveV1(uint)

Packs NFRecord to memory buffer in version 1.0 format.

```
public byte[] SaveV1(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The byte array of packed NFRecord.

Remarks

Note that blocked orientations are not packed by default.

The following flags are supported: FlagSkipSingularPoints, FlagSaveBlockedOrients, FlagSkipCurvatures, FlagSkipGs.

See Also

NFRecord Constructors | Save

5.4.1.2. NFRecord.CoreCollection Class

Represents the collection of NFCore.

```
public sealed class CoreCollection: IList
```

Properties

Capacity	Gets or sets the number of elements that the CoreCollection can contain.

Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the end of the CoreCollection.
Clear	Removes all elements from the CoreCollection.
`CopyTo`	Copies the CoreCollection or a portion of it to a onedimensional array.
`GetEnumerator`	Returns an enumerator that can be used to iterate through the CoreCollection.
Insert	Inserts an element into the CoreCollection at the specified index.
RemoveAt	Removes the element at the specified index within the `NFRecord.CoreCollection`.

5.4.1.2.1. Capacity Property

Gets or sets the number of elements that the CoreCollection can contain.

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

Property value

The number of elements that the `NFRecord.CoreCollection` can contain.

See Also

Count

5.4.1.2.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection.

5.4.1.2.3. NFRecord.CoreCollection.Item Property

Gets or sets the member from collection by index.

```
public NFCore this[
        int index
] {get; set;}
```

Parameters

index	The index of the NFCore structure to retrieve from the collection.

Property value

A NFCore structure.

See also

NFCore Count

5.4.1.2.4. Add Method

Adds an object to the end of the CoreCollection.

```
public int Add(
        NFCore value
);
```

Parameters

value	The NFCore to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

5.4.1.2.5. Clear Method

Removes all elements from the CoreCollection.

```
public void Clear();
```

See Also

RemoveAt

5.4.1.2.6. CopyTo Method

Copies the CoreCollection or a portion of it to a onedimensional array.

```
public void CopyTo(Array array, int index)
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

5.4.1.2.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the CoreCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the checked index collection.

See Also

NFRecord.CoreCollection

5.4.1.2.8. Insert Method

Inserts an element into the CoreCollection at the specified index.

```
public void Insert(
        int index,
        NFCore value
);
```

Parameters

index	The zero-based index location where the NFCore is inserted.
value	The NFCore to add to the collection.

See Also

Add

5.4.1.2.9. RemoveAt Method

Removes the element at the specified index within the `NFRecord.CoreCollection`.

```
public void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the NFCore to remove.

See Also

Clear

5.4.1.3. NFRecord.DeltaCollection Class

Represents the collection of NFDelta.

```
public sealed class DeltaCollection: IList
```

Properties

Capacity	Gets or sets the number of elements that the FmrFingerView.MinutiaCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the NFRecord.DeltaCollection.
Clear	Removes all elements from the NFRecord.DeltaCollection.
CopyTo	Copies the DeltaCollection or a portion of it to a onedimensional array.
GetEnumerator	Returns an enumerator that can be used to iterate through the DeltaCollection.
Insert	Inserts an element into the NFRecord.DeltaCollection at the specified index.
RemoveAt	Removes the element at the specified index within the NFRecord.DeltaCollection.

5.4.1.3.1. Capacity Property

Gets or sets the number of elements that the FmrFingerView.MinutiaCollection can contain.

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

Property value

The number of elements that the NFRecord.DeltaCollection can contain.

See Also

Count

5.4.1.3.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection.

5.4.1.3.3. NFRecord.DeltaCollection.Item Property

Gets or sets the member from collection by index.

```
public NFDelta this[
        int index
] {get; set;}
```

Parameters

index	The index of the NFDelta structure to retrieve from the collection.

Property value

A NFDelta structure.

See also

Count

5.4.1.3.4. Add Method

Adds an object to the `NFRecord.DeltaCollection`.

```
public int Add(
        NFDelta value
);
```

Parameters

value	The NFDelta to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

5.4.1.3.5. Clear Method

Removes all elements from the `NFRecord.DeltaCollection`.

```
public virtual void Clear();
```

See Also

RemoveAt

5.4.1.3.6. CopyTo Method

Copies the DeltaCollection or a portion of it to a onedimensional array.

```
public void CopyTo(Array array, int index)
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

5.4.1.3.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the DeltaCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the core collection.

5.4.1.3.8. Insert Method

Inserts an element into the `NFRecord.DeltaCollection` at the specified index.

```
public void Insert(
        int index,
        NFDelta value
);
```

Parameters

index	The zero-based index location where the NFDelta is inserted.
value	The NFDelta to add to the collection.

See Also

Add

5.4.1.3.9. RemoveAt Method

Removes the element at the specified index within the `NFRecord.DeltaCollection`.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the NFDelta to remove.

See Also

Clear

5.4.1.4. NFRecord.DoubleCoreCollection Class

Represents the collection of NFDoubleCore.

```
public sealed class DoubleCoreCollection: IList
```

Properties

Capacity	Gets or sets the number of elements that the NFRecord.DoubleCoreCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the `NFRecord.DoubleCoreCollection`.
Clear	Removes all elements from the `NFRecord.DoubleCoreCollection`.

CopyTo	Copies the DoubleCoreCollection or a portion of it to a onedimensional array.
GetEnumerator	Returns an enumerator that can be used to iterate through the DoubleCoreCollection.
Insert	Inserts an element into the NFRecord.DoubleCoreCollection at the specified index.
RemoveAt	Removes the element at the specified index of the NFRecord.DoubleCoreCollection.

5.4.1.4.1. Capacity Property

Gets or sets the number of elements that the NFRecord.DoubleCoreCollection can contain.

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

Property value

The number of elements that the NFRecord.DoubleCoreCollection can contain.

See Also

Count

5.4.1.4.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection.

5.4.1.4.3. Item Property

Gets or sets the member from collection by index.

```
public NFDoubleCore this[
        int index
        ] {get; set;}
```

Property value

A NFDoubleCore structure.

See also

NFDoubleCore Count

5.4.1.4.4. Add Method

Adds an object to the `NFRecord.DoubleCoreCollection`.

```
public int Add(
        NFDoubleCore value
);
```

Parameters

value	The NFDoubleCore to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

5.4.1.4.5. Clear Method

Removes all elements from the `NFRecord.DoubleCoreCollection`.

```
public virtual void Clear();
```

See Also

RemoveAt

5.4.1.4.6. CopyTo Method

Copies the DoubleCoreCollection or a portion of it to a onedimensional array.

```
public void CopyTo(Array array, int index)
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.

index	The zero-based index in array at which copying begins.

5.4.1.4.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the DoubleCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the double core collection.

5.4.1.4.8. Insert Method

Inserts an element into the `NFRecord.DoubleCoreCollection` at the specified index.

```
public void Insert(
        int index,
        NFDoubleCore value
);
```

Parameters

index	The zero-based index location where the NFDoubleCore is inserted.
value	The NFDoubleCore to add to the collection.

See Also

Add

5.4.1.4.9. RemoveAt Method

Removes the element at the specified index of the `NFRecord.DoubleCoreCollection`.

```
public virtual void RemoveAt(int index);
```

Parameters

index	The zero-based index of the NFDoubleCore to remove.

Copyright © 2006 Neurotechnologija

See Also

Clear

5.4.1.5. NFRecord.MinutiaCollection Class

Represents the collection of NFMinutia.

```
public sealed class MinutiaCollection: IList
```

Properties

Capacity	Gets or sets the number of elements that the NFRecord.MinutiaCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the `NFRecord.MinutiaCollection`.
Clear	Removes all elements from the `NFRecord.MinutiaCollection`.
`CopyTo`	Copies the MinutiaCollection or a portion of it to a one- dimensional array.
`GetEnumerator`	Returns an enumerator that can be used to iterate through the MinutiaCollection.
Insert	Inserts an element into the `NFRecord.MinutiaCollection` at the specified index.
RemoveAt	Removes the element at the specified index within the `NFRecord.MinutiaCollection`.

5.4.1.5.1. Capacity Property

Gets or sets the number of elements that the NFRecord.MinutiaCollection can contain.

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

Property value

The number of elements that the `NFRecord.MinutiaCollection` can contain.

See Also

Count

5.4.1.5.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection.

5.4.1.5.3. MinutiaCollection.Item Property

Gets or sets the member from collection by index.

```
public NFMinutia this[
        int index
] {get; set;}
```

Parameters

index	The index of the NFMinutia structure to retrieve from the collection.

Property value

A NFMinutia structure.

See also

NFMinutia | Count

5.4.1.5.4. Add Method

Adds an object to the `NFRecord.MinutiaCollection`.

```
public int Add(
        NFMinutia value
);
```

Parameters

value	The NFMinutia to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

5.4.1.5.5. Clear Method

Removes all elements from the `NFRecord.MinutiaCollection`.

```
public virtual void Clear()
```

See Also

RemoveAt

5.4.1.5.6. CopyTo Method

Copies the MinutiaCollection or a portion of it to a one- dimensional array.

```
public void CopyTo(Array array, int index)
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

5.4.1.5.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the MinutiaCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the minutia collection.

5.4.1.5.8. Insert Method

Inserts an element into the `NFRecord.MinutiaCollection` at the specified index.

```
public void Insert(
        int index,
        NFMinutia value
);
```

Parameters

index	The zero-based index location where the NFMinutia is inserted.
value	The NFMinutia to add to the collection.

See Also

Add

5.4.1.5.9. RemoveAt Method

Removes the element at the specified index within the `NFRecord.MinutiaCollection`.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the NFMinutia to remove.

See Also

Clear

5.4.1.6. NFRecord.MinutiaNeighboursCollection Class

Represents the collection of NFMinutiaNeighbour.

Properties

Count	Gets or sets the number of elements that the MinutiaNeighboursCollection can contain.
Item	Gets the member of items in the collection.

Methods

GetCount	Retrieves the number of minutia neighbours at the specified index.
GetEnumerator	Returns an enumerator that can be used to iterate through the MinutiaNeighboursCollection.

5.4.1.6.1. Count Property

Gets or sets the number of elements that the MinutiaNeighboursCollection can contain.

```
public int Count {get;}
```

Property value

The number of items in the collection

5.4.1.6.2. NFRecord.MinutiaNeighboursCollection.Item Property

5.4.1.6.2.1. this[int]

Gets the members array from collection by index;

```
public NFMinutiaNeighbour[] this[
                int minutiaIndex
        ] {get;}
```

Parameters

minutiaIndex	The index of the specified minutia.

Property value

A NFMinutiaNeighbour structures array of specified minutia.

See Also

NFMinutiaNeighbour | Count

5.4.1.6.2.2. this[int, int]

Gets or sets the member from collection by index

```
public NFMinutiaNeighbour this[
                int minutiaIndex,
                int index
        ] {get; set;}
```

Parameters

minutiaIndex	The index of the specified minutia.
index	The index of the specified neighbour minutia to retrieve from the collection.

Property value

A NFMinutiaNeighbour structure of specified neighbour minutia.

See Also

NFMinutiaNeighbour | >Count

5.4.1.6.3. GetCount Method

Retrieves the number of minutia neighbours at the specified index.

```
public int GetCount(
        int minutiaIndex
);
```

Parameters

minutiaIndex	The index of the minutia.

Return Values

The number of the minutia neighbours.

5.4.1.6.4. GetEnumerator Method

Returns an enumerator that can be used to iterate through the MinutiaNeighboursCollection.

Copyright © 2006 Neurotechnologija

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the minutia neighbours collection.

5.4.1.7. NFCore Struct

The structure contains information of core.

```
public struct NFCore
```

Constructors

NFCore Constructor	Initializes a new instance of the `NFCore` structure.

Properties

Angle	Gets or sets Angle of core.
RawAngle	Gets or sets raw angle of core
X	Gets or sets y coordinate of core.
Y	Gets or sets x coordinate of core.

5.4.1.7.1. NFCore Constructor

5.4.1.7.1.1. NFCore(ushort, ushort)

Initializes a new instance of the NFCore structure.

```
public NFCore(
        ushort x,
        ushort y
);
```

Parameters

ushort	The x - coordinate of the core.
ushort	The y - coordinate of the core.

5.4.1.7.1.2. NFCore(ushort x, ushort y, int angle)

```
public NFCore(
        ushort x,
        ushort y,
        int angle
);;
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.
angle	The angle of the core.

5.4.1.7.1.3. NFCore(ushort x, ushort y, double angle)

```
public NFCore(
        ushort x,
        ushort y,
        double angle
);
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.
angle	The angle of the core.

5.4.1.7.2. Angle Property

Gets or sets Angle of core.

```
public double Angle {get; set;}
```

Property value

The angle of the core.

5.4.1.7.3. RawAngle Property

Gets or sets raw angle of core

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

Property value

The raw angle of the core.

Remarks

The angle of the core is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the angle of the core is unknown.

5.4.1.7.4. X Property

Gets or sets y coordinate of core.

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

Property value

The y coordinate of the core.

Remarks

The x coordinate of the core is specified in pixels at Resolution and X * [NFRecord horizontal resolution] / `Resolution` can not be greater than MaxDimension or NFRecord width minus one.

5.4.1.7.5. Y Property

Gets or sets x coordinate of core.

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

Property value

The y coordinate of the core.

Remarks

The y coordinate of the core is specified in pixels at Resolution and Y * [NFRecord vertical resolution] / `Resolution` can not be greater than MaxDimension or NFRecord width minus one.

5.4.1.8. NFDelta Struct

The structure contains information of delta.

```
public struct NFDelta
```

Constructors

NFDelta Constructor	Initializes a new instance of the NFDelta structure.

Properties

Angle1	Gets or sets the first angle of delta.
Angle2	Gets or sets the second angle of delta.
Angle3	Gets or sets the third angle of delta.
RawAngle1	Gets or sets the first row angle of the delta.
RawAngle2	Gets or sets the second row angle of the delta.
RawAngle3	Gets or sets the third row angle of the delta.
X	Gets or sets x coordinate of delta.
Y	Gets or sets y coordinate of delta.

5.4.1.8.1. NFDelta Constructor

Initializes a new instance of the NFDelta structure.

5.4.1.8.1.1. NFDelta(ushort x, ushort y)

```
public NFDelta(
        ushort x,
        ushort y
);
```

Parameters

x	The x - coordinate of the delta.
y	The y - coordinate of the delta.

5.4.1.8.1.2. NFDelta(ushort x, ushort y, int angle1, int angle2, int angle3)

```
public NFDelta(
        ushort x,
        ushort y,
        int angle1,
        int angle2,
        int angle3
);
```

Parameters

x	The x - coordinate of the delta.
y	The y - coordinate of the delta.
angle1	The first angle of the delta.
angle2	The second angle of the delta.
angle3	The third angle of the delta.

5.4.1.8.1.3. NFDelta(ushort x, ushort y, double angle1, double angle2, double angle3)

```
public NFDelta(
        ushort x,
        ushort y,
        double angle1,
        double angle2,
        double angle3
);
```

Parameters

x	The x - coordinate of the delta.
y	The y - coordinate of the delta.
angle1	The first angle of the delta.
angle2	The second angle of the delta.
angle3	The third angle of the delta.

5.4.1.8.2. Angle1 Property

Gets or sets the first angle of delta.

```
public double Angle1 {get; set;}
```

Property value

The first angle of the delta.

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

5.4.1.8.3. Angle2 Property

Gets or sets the second angle of delta.

```
public double Angle2 {get; set;}
```

Property value

The second angle of the delta.

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

5.4.1.8.4. Angle3 Property

Gets or sets the third angle of delta.

```
public double Angle3 {get; set;}
```

Property value

The third angle of the delta.

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

5.4.1.8.5. RawAngle1 Property

```
public int RawAngle1 {get; set;}
```

Property value

The first raw angle of the delta.

5.4.1.8.6. RawAngle2 Property

```
public int RawAngle2 {get; set;}
```

Property value

The second raw angle of the delta.

5.4.1.8.7. RawAngle3 Property

```
public int RawAngle3 {get; set;}
```

Property value

The third raw angle fog the delta.

5.4.1.8.8. X Property

Gets or sets x coordinate of delta.

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

Property value

The x coordinate of the delta.

Remarks

The x coordinate of the delta is specified in pixels at Resolution and X * [NFRecord horizontal resolution] / `Resolution` can not be greater than MaxDimension or NFRecord width minus one.

5.4.1.8.9. Y Property

Gets or sets y coordinate of delta.

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

Property value

The x coordinate of the delta.

Remarks

The y coordinate of the delta is specified in pixels at Resolution and Y * [NFRecord vertical resolution] / `Resolution` can not be greater than MaxDimension or NFRecord width minus one.

5.4.1.9. NFDoubleCore Struct

The structure contains information of double core.

```
public struct NFDoubleCore
```

Constructors

NFDoubleCore Constructor	Initializes a new instance of the `NFDoubleCore` structure.

Properties

X	Gets or sets x coordinate of the double core.
Y	Gets or sets y coordinate of the double core.

5.4.1.9.1. NFDoubleCore Constructor

```
public NFDoubleCore(ushort x, ushort y);
```

Parameters

x	The x - coordinate of the double core.
y	The y - coordinate of the double core.

5.4.1.9.2. X Property

Gets or sets x coordinate of the double core.

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

Property value

The X coordinate of the double core.

5.4.1.9.3. Y Property

Gets or sets y coordinate of the double core.

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

Property value

The Y coordinate of the double core.

5.4.1.10. NFMinutia Struct

The structure contains information of minutia.

```
public struct NFMinutia
```

Constructors

NFMinutia Constructor	Initializes a new instance of the NFMinutia structure.

Properties

Angle	Gets or sets the angle of the minutia.
Curvature	Gets or sets the ridge curvature near minutia.
G	Gets or sets the G (ridge density) near minutia.
Quality	Gets or sets quality of the minutia.
RawAngle	Gets or sets the raw angle of the minutia.
Type	Gets or sets the type of the minutia.
X	Gets or sets x coordinate of the minutia.
Y	Gets or sets y coordinate of the minutia.

5.4.1.10.1. NFMinutia Constructor

Initializes a new instance of the NFMinutia structure.

5.4.1.10.1.1. NFMinutia(ushort x, ushort y, NFMinutiaType type, byte angle)

```
public NFMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        byte angle
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

5.4.1.10.1.2. NFMinutia(ushort x, ushort y, NFMinutiaType type, double angle)

```
public NFMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        double angle
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

5.4.1.10.1.3. NFMinutia(ushort x, ushort y, NFMinutiaType type, byte angle, byte quality, byte curvature, byte g)

```
public NFMinutia(
        ushort x,
```

```
        ushort y,
        NFMinutiaType type,
        byte angle,
        byte quality,
        byte curvature,
        byte g
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.
quality	The quality of the minutia.
curvature	The ridge curvature near minutia.
g	The G (ridge density) near minutia.

5.4.1.10.1.4. NFMinutia(ushort x, ushort y, NFMinutiaType type, double angle, byte quality, byte curvature, byte g)

```
public NFMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        double angle,
        byte quality,
        byte curvature,
        byte g
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.
quality	The quality of the minutia.

curvature	The ridge curvature near minutia.
g	The G (ridge density) near minutia.

5.4.1.10.2. Angle Property

Gets or sets the angle of the minutia.

```
public double Angle {get; set;}
```

Property value

The angle of the minutia.

Remarks

The angle of the minutia is specified in 180/128 degrees units in counterclockwise order and can not be greater than 256 minus one.

5.4.1.10.3. Curvature Property

Gets or sets the ridge curvature near minutia.

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

Property value

The ridge curvature near minutia.

Remarks

If curvature of the minutia is unknown it must be set to 255.

5.4.1.10.4. G Property

Gets or sets the G (ridge density) near minutia.

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

Property value

The G (ridge density) near minutia.

Remarks

If G of the minutia is unknown it must be set to 255.

5.4.1.10.5. Quality Property

Gets or sets quality of the minutia.

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

Property value

The quality of the minutia.

Remarks

The quality of the minutia must be in the range [0, 100]. The higher it is, the better the quality of the minutia is.

If quality of the minutia is unknown it must be set to zero.

5.4.1.10.6. RawAngle Property

Gets or sets the raw angle of the minutia.

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

Property value

The raw angle of the minutia.

5.4.1.10.7. Type Property

Gets or sets the type of the minutia.

```
public NFMinutiaType Type {get; set;}
```

Property value

One of the NFMinutiaType values.

5.4.1.10.8. X Property

Gets or sets x coordinate of the minutia.

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

Property value

The X coordinate of the minutia.

Copyright © 2006 Neurotechnologija

Remarks

The x coordinate of the minutia is specified in pixels at `Resolution` and *X* * [NFRecord horizontal resolution] / `Resolution` can not be greater than `MaxDimension` or NFRecord width minus one.

5.4.1.10.9. Y Property

Gets or sets y coordinate of the minutia.

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

Property value

The Y coordinate of the minutia.

Remarks

The y coordinate of the minutia is specified in pixels at `Resolution` and *Y* * [NFRecord vertical resolution] / `Resolution` can not be greater than `MaxDimension` or NFRecord width minus one.

5.4.1.11. NFMinutiaNeighbour Struct

The structure contains information of minutia neighbour.

```
public struct NFMinutiaNeighbour
```

Constructors

NFMinutiaNeighbour Constructor	Initializes a new instance of the `NFMinutiaNeighbour` structure.

Fields

Empty	Represents a minutia neighbour that is null reference.

Properties

Index	Gets or sets the index of minutia neighbour.
RidgeCount	Gets or sets the ridge count between the minutia and minutia neighbour.

5.4.1.11.1. NFMinutiaNeighbour Constructor

```
public NFMinutiaNeighbour(
        int index,
        byte ridgeCount
);
```

Parameters

index	The index of neighbour.
ridgeCount	The ridge count of neighbour.

5.4.1.11.2. Empty Field

Represents a NFMinutiaNeighbour that is a null reference.

```
public static readonly NFMinutiaNeighbour Empty;
```

5.4.1.11.3. Index Property

Gets or sets the index of minutia neighbour.

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

Property value

The index of minutia neighbour.

5.4.1.11.4. RidgeCount Property

Gets or sets the ridge count between the minutia and minutia neighbour.

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

Property value

The ridge count between the minutia and minutia neighbour.

5.4.1.12. NFImpressionType Enumeration

Specifies the impression types.

```
public enum NFImpressionType
```

Members

Member name	Description
LatentImpression	Latent impression fingerprint.
LatentLift	Latent lift fingerprint.
LatentPhoto	Latent photo fingerprint.
LatentTracing	Latent tracing fingerprint.
LiveScanContactless	Live-scanned fingerprint using contactless device.
LiveScanPlain	Live-scanned plain fingerprint.
LiveScanRolled	Live-scanned rolled fingerprint.
NonliveScanPlain	Nonlive-scanned (from paper) plain fingerprint.
NonliveScanRolled	Nonlive-scanned (from paper) rolled fingerprint.
Swipe	Live-scanned fingerprint by sliding the finger across a "swipe" sensor.

5.4.1.13. NFMinutiaFormat Enumeration

Specifies formats of minutia. This enumeration allows a bitwise combination of its member values.

```
public enum NFMinutiaFormat
```

Members

Member name	Description
HasCurvature	If near minutia is ridge curvature
HasG	If near minutia is G(ridge density).
HasQuality	The quality of the minutia.

5.4.1.14. NFMinutiaType Enumeration

Specifies types of minutia.

```
public enum NFMinutiaType
```

Members

Member name	Description
Bifurcation	The minutia that is a bifurcation of a ridge.
End	The minutia that is an end of a ridge.
Unknown	The type of the minutia is unknown.

5.4.1.15. NFPatternClass Enumeration

Specifies pattern class of the fingerprint.

```
public enum NFPatternClass
```

Members

Member name	Description
AccidentalWhorl	Accidental whorl pattern class.
Amputation	Amputation. Pattern class is not available.
CentralPocketLoop	Central pocket loop pattern class.
DoubleLoop	Double loop pattern class.
LeftSlantLoop	Left slant loop pattern class.
PlainArch	Plain arch pattern class.
PlainWhorl	Plain whorl pattern class.
RadialLoop	Radial loop pattern class.
RightSlantLoop	Right slant loop pattern class.
Scar	Scar. Pattern class is not available.
TentedArch	Tented arch pattern class.
UlnarLoop	Ulnar loop pattern class.
Unknown	Unknown pattern class.

Copyright © 2006 Neurotechnologija

Member name	Description
Whorl	Whorl pattern class.

Remarks

This enumeration is implemented according to ANSI/NIST-ITL 1-2000 standard.

5.4.1.16. NFPosition Enumeration

Specifies finger position.

```
public enum NFPosition
```

Members

Member name	Description
LeftIndex	Index finger of the left hand.
LeftLittle	Little finger of the left hand.
LeftMiddle	Middle finger of the left hand.
LeftRing	Ring finger of the left hand.
LeftThumb	Thumb of the left hand.
RightThumb	Thumb of the right hand.
RightIndex	Index finger of the right hand.
RightLittle	Little finger of the right hand.
RightMiddle	Middle finger of the right hand.
RightRing	Ring finger of the right hand.
Unknown	Unknown finger.

5.4.1.17. NFRidgeCountsType Enumeration

Specifies type of ridge counts contained in NFRecord.

```
public enum NFRidgeCountsType
```

Members

Member name	Description
EightNeighbours	The NFRecord contains ridge counts to closest minutia in each of the eight sectors of each minutia. First sector starts at minutia angle.
EightNeighboursWithIndexes	The NFRecord contains ridge counts to eight neighbours of each minutia.
FourNeighbours	The NFRecord contains ridge counts to closest minutia in each of the four sectors of each minutia. First sector starts at minutia angle.
FourNeighboursWithIndexes	The NFRecord contains ridge counts to four neighbours of each minutia.
None	The NFRecord does not contain ridge counts.
Unspecified	For internal use.

5.5. Neurotec.Biometrics.NFTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates.

DLL: `Neurotec.Biometrics.NFTemplate.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).

5.5.1. Neurotec.Biometrics Namespace

Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).

Classes

Class	Description
NFTemplate	Provides methods for packing, unpacking

Class	Description
	and editing Neurotechnologija Fingers Templates (NFTemplates).
NFTemplate.RecordCollection	Represents the collection of NFRecord.
NFTemplateInfo	For internal use.

5.5.1.1. NFTemplate Class

Constructors

NFTemplate Constructor	Initializes a new instance of the `NFTemplate` class.

Properties

Handle	Gets handle to unmanaged `NFTemplate` object.
Records	Gets Records collection.

Methods

CalculateSize	Calculates `NFTemplate` size.
Check	Checks if format of packed `NFTemplate` format is correct.
Clone	Creates `NFTemplate` object from another `NFTemplate` object.
Dispose	Releases the resources used by NFTemplate.
FromHandle	Creates `NFTemplate` object from handle.
GetRecordCount	Retrieves records count.
GetSize	Calculates packed size of `NFTemplate` object.
Pack	Creates `NFTemplate` object with `NFRecord` objects and packs into byte array.
Save	Packs `NFTemplate` object to byte array.

Copyright © 2006 Neurotechnologija

Unpack	Unpacks `NFTemplate` object.

Constants

DllName	Name of DLL containing unmanaged part of this class.
FlagProcessFirstRecordOnly	The flag indicating whether only the first finger record should be unpacked or packed while unpacking or packing NFTemplate.
MaxRecordCount	The maximum number of finger records NFTemplate can contain.

5.5.1.1.1. NFTemplate Constructor

5.5.1.1.1.1. NFTemplate();

```
public NFTemplate()
```

5.5.1.1.1.2. NFTemplate(byte[] buffer)

Creates NFTemplate object from packed NFTemplate object.

```
public NFTemplate(
        byte[] buffer
);
```

Parameters

buffer	A byte array with packed NFTemplate objects.

See Also

Pack | Save

5.5.1.1.1.3. NFTemplate(byte[] buffer, uint flags)

Creates NFTemplate object from packed NFTemplate object. Behavior is controlled through flags.

```
public NFTemplate(
```

```
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	A byte array with packed NFTemplate objects.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

See Also

Pack | Save

5.5.1.1.1.4. NFTemplate(byte[] buffer, out NFTemplateInfo info)

Creates NFTemplate object from packed NFTemplate object.

```
public NFTemplate(
        byte[] buffer,
        out NFTemplateInfo info
);
```

Parameters

buffer	A byte array with packed NFTemplate objects.
info	NFTemplateInfo object.

See Also

Pack | Save

5.5.1.1.1.5. NFTemplate(byte[] buffer, uint flags, out NFTemplateInfo info)

Creates NFTemplate object from packed NFTemplate object. Behavior is controlled through flags.

```
public NFTemplate(
        byte[] buffer,
        uint flags,
        out NFTemplateInfo info
);
```

Parameters

buffer	A byte array with packed NFTemplate objects.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
info	NFTemplateInfo object.

See Also

Pack | Save

5.5.1.1.2. Handle Property

Gets handle to unmanaged `NFTemplate` object.

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

Property value

A handle to unmanaged `NFTemplate` object.

See Also

FromHandle

5.5.1.1.3. Records Property

Gets Records collection.

```
public NFTemplate.RecordCollection Records {get;}
```

Property value

A NFTemplate.RecordCollection collection.

See Also

NFTemplate.RecordCollection | NFRecord

5.5.1.1.4. CalculateSize Method

Calculates `NFTemplate` size.

```
public static int CalculateSize(
        int[] recordSizes
```

```
);
```

Parameters

recordSizes	An array that contains `NFRecords` sizes.

Return Values

A size of records.

See Also

NFRecord

5.5.1.1.5. Check Method

Checks if format of packed `NFTemplate` format is correct.

5.5.1.1.5.1. Check(byte[] buffer)

Checks if format of packed `NFTemplate` format is correct.

```
public static void Check(
        byte[] buffer
);
```

Parameters

buffer	A byte array with packed `NFTemplate`.

Return Values

`true` if `NFTemplate` format is correct, `false` otherwise.

5.5.1.1.5.2. Check(IntPtr buffer, int bufferSize)

Checks if format of packed `NFTemplate` format is correct.

```
public static void Check(
        IntPtr buffer,
        int bufferSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFTemplate`.
bufferSize	Size of memory buffer that contains packed `NFTemplate`.

Return Values

`true` if `NFTemplate` format is correct, `false` otherwise.

5.5.1.1.6. Clone Method

Creates `NFTemplate` object from another `NFTemplate` object.

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

Return Values

A NFTemplate object.

See Also

NFTemplate

5.5.1.1.7. Dispose Method

Releases the resources used by NFTemplate.

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

See Also

NFTemplate

5.5.1.1.8. FromHandle Method

Creates `NFTemplate` object from handle.

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

Parameters

handle	Handle to unmanaged `NFTemplate` object.

Return Values

A NFTemplate object.

See Also

NFTemplate | Handle

5.5.1.1.9. GetRecordCount Method

Retrieves records count.

5.5.1.1.9.1. GetRecordCount(byte[] buffer)

Retrieves records count.

```
public static int GetRecordCount(
        byte[] buffer
);
```

Parameters

buffer	A byte array with `NFTemplate`.

Return Values

Records count.

See Also

NFRecord

5.5.1.1.9.2. GetRecordCount(IntPtr buffer, int bufferSize)

Retrieves records count.

```
public static int GetRecordCount(
        IntPtr buffer,
        int bufferSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFTemplate`.
bufferSize	Size of memory buffer that contains packed `NFTemplate`.

Return Values

Records count.

See Also

NFRecord

5.5.1.1.10. GetSize Method

Calculates packed size of `NFTemplate` object.

5.5.1.1.10.1. GetSize()

Calculates packed size of `NFTemplate` object.

```
public int GetSize();
```

Return Values

Size of `NFTemplate` object.

5.5.1.1.10.2. GetSize(uint flags)

Calculates packed size of `NFTemplate` object. Behavior is controlled through flags.

```
public int GetSize(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

Size of `NFTemplate` object.

5.5.1.1.11. Pack Method

Creates `NFTemplate` object with `NFRecord` objects and packs into byte array.

5.5.1.1.11.1. Pack(params byte[][] records)

Packs `NFRecord` objects into byte array.

```
public static byte[] Pack(
```

```
        params byte[][] records
);
```

Parameters

records	A byte array of NFRecord objects byte arrays.

Return Values

A byte array with packed NFTemplate objects.

See Also

NFRecord | Save

5.5.1.1.11.2. Pack(IntPtr[] records, int[] recordSizes)

Packs NFRecord objects into byte array from memory buffer.

```
public static byte[] Pack(
        IntPtr[] records,
        int[] recordSizes
);
```

Parameters

records	Pointer to NFRecord objects array.
recordSizes	An array that contains NFRecord objects sizes.

Return Values

A byte array with packed NFTemplate object.

See Also

NFRecord | Save

5.5.1.1.12. Save Method

Packs NFTemplate object to byte array.

5.5.1.1.12.1. Save()

Saves `NFTemplate` object to byte array.

```
public byte[] Save();
```

Return Values

A byte array with packed `NFTemplate` object.

See Also

Pack

5.5.1.1.12.2. Save(uint flags)

Packs `NFTemplate` object to byte array. Behavior is controlled through flags.

```
public byte[] Save(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that control behavior of the method.

Return Values

A byte array with packed `NFTemplate` object.

See Also

Pack

5.5.1.1.13. Unpack Method

Unpacks `NFTemplate` object.

5.5.1.1.13.1. int Unpack(IntPtr buffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr[] records, out int[] recordSizes)

Unpacks `NFTemplate` object from memory buffer.

```
public static int Unpack(
        IntPtr buffer,
        int bufferSize,
        out byte majorVersion,
        out byte minorVersion,
```

```
        out uint size,
        out byte headerSize,
        out IntPtr[] records,
        out int[] recordSizes
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFTemplate` object.
bufferSize	Size of memory buffer that contains packed `NFTemplate`.
majorVersion	Major version of `NFTemplate`.
minorVersion	Minor version of `NFTemplate`.
size	The size of buffer from header of packed NFTemplate. This size is equal to buffer-Size.
headerSize	The size of packed NFTemplate header size.
records	The array of NFRecords.
recordSizes	An array that contains `NFRecord` sizes.

Return Values

The recourds count.

See Also

NFRecord | Pack

5.5.1.1.13.2. void Unpack(IntPtr buffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[][] records)

Unpacks `NFTemplate` object from memory buffer.

```
public static void Unpack(
        IntPtr buffer,
        int bufferSize,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out byte[][] records
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFTemplate` object.
bufferSize	Size of memory buffer that contains packed `NFTemplate`.
majorVersion	Major version of `NFTemplate`.
minorVersion	Minor version of `NFTemplate`.
size	The size of buffer from header of packed NFTemplate. This size is equal to buffer-Size.
headerSize	The size of packed NFTemplate header size.
records	A byte array that contains `NFRecord` objects bytes arrays.

See Also

NFRecord | Pack

5.5.1.1.13.3. int Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr[] records, out int[] recordSizes)

Unpacks `NFTemplate` object from byte array.

```
public static int Unpack(
        byte[] buffer,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out IntPtr[] records,
        out int[] recordSizes
);
```

Parameters

buffer	Byte array that contains packed `NFTemplate` object.
majorVersion	Major version of `NFTemplate`.
minorVersion	Minor version of `NFTemplate`.

size	The size of buffer from header of packed NFTemplate. This size is equal to buffer-Size.
headerSize	The size of packed NFTemplate header size.
records	The array of NFRecords.
recordSizes	An array that contains `NFRecord` sizes.

Return Values

The recourds count.

See Also

NFRecord | Pack | Save

5.5.1.1.13.4. void Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[][] records)

Unpacks `NFTemplate` object from byte array.

```
public static void Unpack(
        byte[] buffer,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out byte[][] records
);
```

Parameters

buffer	Byte array that contains packed `NFTemplate` object.
majorVersion	Major version of `NFTemplate`.
minorVersion	Minor version of `NFTemplate`.
size	The size of buffer from header of packed NFTemplate. This size is equal to buffer-Size.
headerSize	The size of packed NFTemplate header size.
records	A byte array that contains `NFRecord` objects bytes arrays.

Copyright © 2006 Neurotechnologija

See Also

NFRecord | Pack | Save

5.5.1.2. NFTemplate.RecordCollection Class

Properties

Capacity	Gets or sets the number of elements that the RecordCollection can contain.
Item	Gets the member from collection by index.

Methods

Add	Adds an object to the end of the RecordCollection.
AddCopy	Copies NFRecord object to the NFTemplate.RecordCollection.

5.5.1.2.1. Capacity Property

Gets or sets the number of elements that the RecordCollection can contain.

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

Property value

The number of elements that the `NFTemplate.RecordCollection` can contain.

5.5.1.2.2. NFTemplate.RecordCollection.Item Property

Gets the member from collection by index.

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

Property value

A NFRecord object.

5.5.1.2.3. Add Methods

Adds an object to the end of the RecordCollection.

5.5.1.2.3.1. Add(byte[] buffer)

```
public NFRecord Add(
        byte[] buffer
);
```

Parameters

buffer	The packed NFRecord object.

Return Values

The added to the collection NFRecord object.

See Also

NFRecord

5.5.1.2.3.2. Add(byte[] buffer, uint flags)

```
public NFRecord Add(
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	The packed NFRecord object.
flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The added to the collection NFRecord object.

See Also

NFRecord

5.5.1.2.3.3. Add(ushort width, ushort height, ushort horzResolution, ushort vertResolution)

```
public NFRecord Add(
        ushort width,
        ushort height,
        ushort horzResolution,
        ushort vertResolution
);
```

Parameters

width	The width of fingerprint image.
height	The height of fingerprint image.
horzResolution	The horizontal resolution of fingerprint image.
vertResolution	The vertical resolution of fingerprint image.

Return Values

The added to the collection NFRecord object.

See Also

NFRecord

5.5.1.2.3.4. Add(ushort,ushort,ushort,ushort,uint)

```
public NFRecord Add(
        ushort width,
        ushort height,
        ushort horzResolution,
        ushort vertResolution,
        uint flags
);
```

Parameters

width	The width of fingerprint image.
height	The height of fingerprint image.
horzResolution	The horizontal resolution of fingerprint image.
vertResolution	The vertical resolution of fingerprint image.
flags	Bitwise combination of zero or more flags

	that controls behavior of the method.

Return Values

The added to the collection NFRecord object.

See Also

NFRecord

5.5.1.2.4. AddCopy Method

Copies NFRecord object to the NFTemplate.RecordCollection.

```
public NFRecord AddCopy(
        NFRecord srcRecord
);
```

Parameters

srcRecord	The NFRecord object.

Return Values

The copied to the collection NFRecord object.

See Also

NFRecord

5.6. Neurotec.Biometrics.NTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

DLL: `Neurotec.Biometrics.NTemplate.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

5.6.1. Neurotec.Biometrics Namespace

Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

Classes

Class	Description
NTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Templates (NTemplates).
NTemplateInfo	For internal use.

5.6.1.1. NTemplate Class

Constructors

NTemplate Constructor	Initializes a new instance of the `NTemplate` class.

Properties

Fingers	Gets NFTemplate object.
Handle	Gets handle to unmanaged `NTemplate` object.

Methods

AddFingers	Creates new NFTemplate object.
AddFingersCopy	Creates one NFTemplate object from another.
CalculateSize	Calculates `NTemplate` size.
Check	Checks if format of packed `NTemplate` format is correct.
Clear	Removes all `NFRecord` objects.
Clone	Creates `NTemplate` object from another `NTemplate` object.
Dispose	Releases the resources used by `NTemplate`.

Copyright © 2006 Neurotechnologija

FromHandle	Creates `NTemplate` object from handle.
GetSize	Calculates packed size of NFTemplate object.
Pack	Packs packed fingers template as NTemplate into the specified memory buffer.
RemoveFingers	Removes fingers template from the NTemplate.
Save	Packs `NTemplate` object to byte array.
Unpack	Unpacks packed fingers template from the packed NTemplate.

Constants

DllName	Name of DLL containing unmanaged part of this class.

5.6.1.1.1. NTemplate Constructor

Initializes a new instance of the `NTemplate` class.

5.6.1.1.1.1. NTemplate()

Initializes a new instance of the `NTemplate` class.

```
public NTemplate();
```

5.6.1.1.1.2. NTemplate(byte[] buffer)

Initializes a new instance of the `NTemplate` class from byte array.

```
public NTemplate(
        byte[] buffer
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.

See Also

Pack | Save

5.6.1.1.1.3. NTemplate(byte[] buffer, uint flags)

Initializes a new instance of the `NTemplate` class from byte array. Behavior is controlled through flags.

```
public NTemplate(
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

See Also

Pack | Save

5.6.1.1.1.4. NTemplate(byte[] buffer, out NTemplateInfo info)

Initializes a new instance of the `NTemplate` class from byte array.

```
public NTemplate(
        byte[] buffer,
        out NTemplateInfo info
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.
info	The `FMRecordInfo` object.

See Also

Pack | Save

5.6.1.1.1.5. NTemplate(byte[] buffer, uint flags, out NTemplateInfo info)

Initializes a new instance of the `NTemplate` class from byte array. Behavior is controlled through flags.

```
public NTemplate(
        byte[] buffer,
        uint flags,
        out NTemplateInfo info
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
info	The `FMRecordInfo` object.

See Also

Pack | Save

5.6.1.1.2. Fingers Property

Gets NFTemplate object.

```
public NFTemplate Fingers {get;}
```

Property value

A NFTemplate object.

See Also

NFTemplate

5.6.1.1.3. Handle Property

Gets handle to unmanaged `NTemplate` object.

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

Property value

A handle to unmanaged `NTemplate` object.

See Also

FromHandle

5.6.1.1.4. AddFingers Methods

Creates new NFTemplate object.

5.6.1.1.4.1. AddFingers()

```
public NFTemplate AddFingers();
```

Return Values

A NFTemplate object.

See Also

NFTemplate

5.6.1.1.4.2. AddFingers(byte[] buffer)

Creates new NFTemplate object from packed NFTemplate object.

```
public NFTemplate AddFingers(
        byte[] buffer
);
```

Parameters

buffer	The packed NFTemplate object.

Return Values

The NFTemplate object.

See Also

NFTemplate

5.6.1.1.4.3. AddFingers(byte[] buffer, uint flags)

Creates new NFTemplate object from packed NFTemplate object.

```
public NFTemplate AddFingers(
```

```
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	The packed NFTemplate object.
flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The NFTemplate object.

See Also

NFTemplate

5.6.1.1.5. AddFingersCopy Method

Creates one NFTemplate object from another.

```
public NFTemplate AddFingersCopy(
        NFTemplate srcFingers
);
```

Parameters

srcFingers	The NFTemplate object.

Return Values

The NFTemplate object.

See Also

NFTemplate

5.6.1.1.6. CalculateSize Method

Calculates `NTemplate` size.

```
public static int CalculateSize(
        int fingersTemplateSize
);
```

Parameters

fingersTemplateSize	Size of packed NFTemplate.

Return Values

The size of packed NTemplate.

See Also

NFTemplate | Pack

5.6.1.1.7. Check

Checks if format of packed `NTemplate` format is correct.

5.6.1.1.7.1. Check(byte[] buffer)

Checks if format of packed `NTemplate` format is correct.

```
public static void Check(
        byte[] buffer
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.

Return Values

`true` if `NTemplate` format is correct; `false` otherwise.

See Also

Pack | Save

5.6.1.1.7.2. Check(IntPtr buffer, int bufferSize)

Checks if format of packed `NTemplate` format is correct.

```
public static void Check(
        IntPtr buffer,
        int bufferSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NTemplate`.
bufferSize	Size of memory buffer that contains packed `NTemplate`.

Return Values

`true` if `NTemplate` format is correct; `false` otherwise.

See Also

Pack | Save

5.6.1.1.8. Clear Method

Removes all `NFRecord` objects.

```
public void Clear();
```

See Also

NFRecord

5.6.1.1.9. Clone Method

Creates `NTemplate` object from another `NTemplate` object.

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

Return Values

A `NTemplate` object.

5.6.1.1.10. Dispose Method

Releases the resources used by `NTemplate`.

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

5.6.1.1.11. FromHandle Method

Creates `NTemplate` object from handle.

```
public static NTemplate FromHandle(
```

```
        IntPtr handle
);
```

Parameters

handle	Handle to unmanaged `NTemplate` object.

Return Values

A NTemplate object.

See Also

Handle

5.6.1.1.12. GetSize Methods

Calculates packed size of NFTemplate object.

5.6.1.1.12.1. GetSize()

Calculates packed size of NFTemplate object.

```
public int GetSize();
```

Return Values

Size of `NTemplate` object.

5.6.1.1.12.2. GetSize(uint flags)

Calculates packed size of NFTemplate object. Behavior is controlled through flags.

```
public int GetSize(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

Size of `NTemplate` object.

5.6.1.1.13. Pack Methods

Packs packed fingers template as NTemplate into the specified memory buffer.

5.6.1.1.13.1. Pack(byte[] fingersTemplate)

```
public static byte[] Pack(
        byte[] fingersTemplate
);
```

Parameters

fingersTemplate	The byte array with packed NFTemplate.

Return Values

A byte array with packed `NTemplate` objects.

See Also

NFTemplate | Save | Unpack

5.6.1.1.13.2. Pack(IntPtr fingersTemplate, int fingersTemplateSize)

```
public static byte[] Pack(
        IntPtr fingersTemplate,
        int fingersTemplateSize
);
```

Parameters

fingersTemplate	Pointer to memory buffer.
fingersTemplateSize	The fingers template size.

Return Values

A byte array with packed `NTemplate` objects.

See Also

NFTemplate | Save | Unpack

5.6.1.1.14. RemoveFingers Method

Removes fingers template from the NTemplate.

```
public void RemoveFingers();
```

See Also

NFTemplate | AddFingers | AddFingersCopy

5.6.1.1.15. Save Methods

Packs `NTemplate` object to byte array.

5.6.1.1.15.1. Save()

Packs `NTemplate` object to byte array.

```
public byte[] Save();
```

Return Values

A byte array with packed `NTemplate` object.

See Also

Pack

5.6.1.1.15.2. Save(uint flags)

Packs `NTemplate` object to byte array. Behavior is controlled through flags.

```
public byte[] Save(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that control behavior of the method.

Return Values

A byte array with packed `NTemplate` object.

See Also

Pack

5.6.1.1.16. Unpack Methods

Unpacks packed fingers template from the packed NTemplate.

5.6.1.1.16.1. Unpack(IntPtr buffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr fingersTemplate, out int fingersTemplateSize)

Unpacks `NTemplate` object from memory buffer.

```
public static void Unpack(
        IntPtr buffer,
        int bufferSize,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out IntPtr fingersTemplate,
        out int fingersTemplateSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NTemplate` object.
bufferSize	Size of memory buffer that contains packed `NTemplate`.
majorVersion	Major version of `NTemplate`.
minorVersion	Minor version of `NTemplate`.
size	The size of buffer from header of packed NTemplate. This size is equal to bufferSize
headerSize	The size of packed NTemplate header size.
fingersTemplate	Pointer to array that contains packed NFTemplate.
fingersTemplateSize	The size of packed NFTemplate.

See Also

NFTemplate | Pack | Save

5.6.1.1.16.2. Unpack(IntPtr buffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[] fingersTem-

plate)

Unpacks `NTemplate` object from memory buffer.

```
public static void Unpack(
        IntPtr buffer,
        int bufferSize,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out byte[] fingersTemplate
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NTemplate` object.
bufferSize	Size of memory buffer that contains packed `NTemplate`.
majorVersion	Major version of `NTemplate`.
minorVersion	Minor version of `NTemplate`.
size	The size of buffer from header of packed NTemplate. This size is equal to bufferSize
headerSize	The size of packed NTemplate header size.
fingersTemplate	The byte array with packed NFTemplate.

See Also

NFTemplate | Pack | Save

5.6.1.1.16.3. Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr fingersTemplate, out int fingersTemplateSize)

Unpacks `NTemplate` object from byte array.

```
public static void Unpack(
        byte[] buffer,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out IntPtr fingersTemplate,
```

```
        out int fingersTemplateSize
);
```

Parameters

buffer	A byte array that contains packed `NTemplate`.
majorVersion	Major version of `NTemplate`.
minorVersion	Minor version of `NTemplate`.
size	The size of buffer from header of packed NTemplate. This size is equal to bufferSize
headerSize	The size of packed NTemplate header size.
fingersTemplate	Pointer to array that contains packed NFTemplate.
fingersTemplateSize	The size of packed NFTemplate.

See Also

NFTemplate | Pack | Save

5.6.1.1.16.4. Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[] fingersTemplate)

Unpacks `NTemplate` object from byte array.

```
public static void Unpack(
        byte[] buffer,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out byte[] fingersTemplate
);
```

Parameters

buffer	A byte array that contains packed `NTemplate`.
majorVersion	Major version of `NTemplate`.
minorVersion	Minor version of `NTemplate`.

size	The size of buffer from header of packed NTemplate. This size is equal to bufferSize
headerSize	The size of packed NTemplate header size.
fingersTemplate	The byte array with packed NFTemplate.

See Also

NFTemplate | Pack | Save

Appendix A. Support Information

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 the technical support (<support@neurotechnologija.com>) for any help on solving the development problems.

Appendix B. Distribution Content

Template Management and Conversion Add-On distribution contains the following folders and files:

bin/Win*/

A directory containing binaries for Microsoft Windows operating system. Currently there is only `Win32_x86` directory for Windows OS running on 32-bit x86 CPU.

`ANTemplate.dll`	ANTemplate library.
`FMRecord.dll`	FMRecord library.
`NCore.dll`	NCore library.
`NFRecord.dll`	NFRecord library.
`NFTemplate.dll`	NFTemplate library.
`NTemplate.dll`	NTemplate library.
`msvcr71.dll`	Microsoft C Runtime Library 7.1.

bin/dotNET/

A directory containing binaries for .NET Framework 1.1 platform.

`Neurotec.dll`	Neurotec library.
`Neuro-tec.Biometrics.ANTemplate.dll`	Neurotec.Biometrics.ANTemplate library.
`Neuro-tec.Biometrics.FMRecord.dll`	Neurotec.Biometrics.FMRecord library.
`Neuro-tec.Biometrics.NFRecord.dll`	Neurotec.Biometrics.NFRecord library.
`Neuro-tec.Biometrics.NFTemplate.dll`	Neurotec.Biometrics.NFTemplate library.
`Neuro-tec.Biometrics.NTemplate.dll`	Neurotec.Biometrics.NTemplate library.

documentation/

A directory containing documentation.

`TMAC Add-On.chm`	This documentation in CHM format.
`TMAC Add-On.pdf`	This documentation in PDF format.

include/

A directory containing header files. It has two subdirectories - `Windows` and `linux` which contain the same files but with appropriate line-ends for Windows and Linux operating system accordingly.

`ANTemplate.h`	Header file for ANTemplate module.
`FMRecord.h`	Header file for FMRecord module.
`FmrFingerView.h`	Header file for FmrFingerView module.
`NCore.h`	Header file for NCore module.
`NErrors.h`	Header file for NErrors module.
`NFRecord.h`	Header file for NFRecord module.
`NFRecordV1.h`	Additional header file for NFRecord module.
`NFTemplate.h`	Header file for NFTemplate module.
`NParameters.h`	Header file for NParameters module.
`NTemplate.h`	Header file for NTemplate module.
`NTypes.h`	Header file for NTypes module.

lib/Win*/

A directory contains library and/or DLL import library files for Microsoft Windows operating system. Currently there is only `Win32_x86` directory for Windows OS running on 32-bit x86 CPU.

`ANTemplate.dll.lib`	Import library for ANTemplate library.
`FMRecord.dll.lib`	Import library for FMRecord library.
`NCore.dll.lib`	Import library for NCore library.
`NFRecord.dll.lib`	Import library for NFRecord library.

NFTemplate.dll.lib	Import library for NFTemplate library.
NTemplate.dll.lib	Import library for NTemplate library.

lib/linux_*/

A directory containing library files for Linux-based operating systems. Currently there is only linux_x86 directory for Linux OS running on 32-bit x86 CPU.

libANTemplate.so	ANTemplate library.
libFMRecord.so	FMRecord library.
libNCore.so	NCore library.
libNFRecord.so	NFRecord library.
libNFTemplate.so	NFTemplate library.
libNTemplate.so	NTemplate library.

Appendix C. Change Log

This appendix lists Template Management and Conversion 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 1.1.0.0

- UPD: NCore library to version 2.0.1.1.
- UPD: Neurotec library to version 2.0.1.1.
- UPD: NFRecord library to version 1.1.0.0.
- UPD: Neurotec.Biometrics.NFRecord library to version 1.1.0.0.
- UPD: NFTemplate library to version 1.1.0.0.
- UPD: Neurotec.Biometrics.NFTemplate library to version 1.1.0.0.
- UPD: NTemplate library to version 1.1.0.0.
- UPD: Neurotec.Biometrics.NTemplate library to version 1.1.0.0.
- UPD: FMRecord library to version 1.1.0.0.
- UPD: Neurotec.Biometrics.FMRecord library to version 1.1.0.0.
- UPD: ANTemplate library to version 1.0.0.2.
- ADD: Change log to documentation

Version 1.0.0.0

Initial release. Contains the following components:

- ANTemplate library version 1.0.0.1.
- FMRecord library version 1.0.2.0.
- NCore library version 2.0.0.0.
- NFRecord library version 1.0.2.0.
- NFTemplate library version 1.0.0.4.
- NTemplate library version 1.0.0.3.
- Neurotec library version 2.0.0.0.
- Neurotec.Biometrics.ANTemplate library version 1.0.0.0.
- Neurotec.Biometrics.FMRecord library version 1.0.2.0.
- Neurotec.Biometrics.NFRecord library version 1.0.2.0.
- Neurotec.Biometrics.NFTemplate library version 1.0.0.2.
- Neurotec.Biometrics.NTemplate library version 1.0.0.1.

C.1. ANTemplate Library

Version 1.0.0.2

- UPD: Minor updates.

Version 1.0.0.1

- UPD: Minor updates.

Version 1.0.0.0

Initial release.

C.2. FMRecord Library

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.2.0

- CHN: Some changes in interface.
- ADD: Support for Neurotechnologija data.

Version 1.0.1.0

- ADD: Four-neighbours ridge count support.
- CHN: Flags definition.

Version 1.0.0.1

- CHN: Flags definition.

Version 1.0.0.0

Initial release.

C.3. NCore Library

Version 2.0.1.1

Copyright © 2006 Neurotechnologija

- CHN: Minor changes.

Version 2.0.1.0

- ADD: NParameters module instead of NMetaTypes module for internal infrastructure support.
- CHN: Infrastructure optimization for 64-bit support.

Version 2.0.0.0

- ADD: A lot of stuff for internal infrastructure support.
- CHN: Some changes in internal infrastructure support.

Version 1.0.0.2

- ADD: NIndexPair structure.

Version 1.0.0.1

- FIX: Minor fixes in headers.

Version 1.0.0.0

Initial release.

C.4. NFRecord Library

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.2.0

- ADD: Packing to version 1.0 functionality.

Version 1.0.1.1

- UPD: Minor updates.

Copyright © 2006 Neurotechnologija

Version 1.0.1.0

- UPD: Made compatible with new functionality of NFExtractor library version 1.0.1.0.
- ADD: Four-neighbours ridge count support.
- CHN: Flags definition.
- FIX: Minor fixes.

Version 1.0.0.2

- FIX: NFSelectMinutiaeNeighbours parameter list.

Version 1.0.0.1

- FIX: NFRecordGetPositionMem, NFRecordGetImpressionTypeMem, NFRecordGetPatternClassMem, NFRecordGetQualityMem, NFRecordGetGMem functions retrieve correct value from packed NFRecord.
- CHN: NFSelectMinutiaeNeighbours takes flags as last parameter.
- ADD: Additional flag.

Version 1.0.0.0

Initial release.

C.5. NFTemplate Library

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.0.4

- UPD: Minor updates.

Version 1.0.0.3

- ADD: NFTemplateGetRecordCountMem function.
- FIX: Minor fixes.

Version 1.0.0.2

- FIX: Minor fixes.

Version 1.0.0.1

- FIX: NFTemplateUnpack function returned error in some cases.

Version 1.0.0.0

Initial release.

C.6. NTemplate Library

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.0.3

- UPD: Minor updates.

Version 1.0.0.2

- FIX: Minor fixes.

Version 1.0.0.1

- FIX: Signature of packed NTemplate.
- FIX: NTemplateUnpack function returned error in some cases.
- FIX: NTemplateUnpack logic.

Version 1.0.0.0

Initial release.

C.7. Neurotec Library

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.

Copyright © 2006 Neurotechnologija

- 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.

C.8. Neurotec.Biometrics.ANTemplate Library

Version 1.0.0.0

Initial release.

C.9. Neurotec.Biometrics.FMRecord Library

Version 1.1.0.0

- ADD: Editing support.

Version 1.0.2.0

- ADD: Neurotechnologija data support.

Version 1.0.1.0

- ADD: Four-neighbours ridge count support.

Version 1.0.0.0

Initial release.

C.10. Neurotec.Biometrics.NFRecord Library

Version 1.1.0.0

- ADD: Editing support.

Version 1.0.2.0

- ADD: Packing to version 1.0 functionality.

Version 1.0.1.1

- UPD: Minor updates.

Version 1.0.1.0

- ADD: Four-neighbours ridge count support.

Version 1.0.0.1

- FIX: NFSelectMinutiaeNeighbours parameter list.

Version 1.0.0.0

Initial release.

C.11. Neurotec.Biometrics.NFTemplate Library

Version 1.1.0.0

- ADD: Editing support.

Version 1.0.0.2

- UPD: Minor updates.

Version 1.0.0.1

- ADD: GetRecordCount static method.

Version 1.0.0.0
Initial release.

C.12. Neurotec.Biometrics.NTemplate Library

Version 1.1.0.0

- ADD: Editing support.

Version 1.0.0.1

- UPD: Minor updates.

Version 1.0.0.0
Initial release.