Free Fingerprint Verification SDK

Developer's Guide

Table of Contents

Free Fingerprint Verification SDK

1 Preface

The brief information about this developer guide.

Version: 1.0.0.2

Release date: 2010-04-20

1.1 Copyright Notice

1.2 Questions

After you have read this developer's guide and the FAQ (see page 81) chapter and still have more questions or face troubleshoots in using the Free Fingerprint Verification SDK, please feel free to contact us.

Contacts

- **Email**: freesdk@neurotechnology.com. When writing by email thoroughly describe a problem you face. Also you can attach additional files (eg. screenshots, fingerprint images) which can help solve your problem.

1.3 Feedback

The information in this guide has been thoroughly reviewed and tested, but if you have noticed errors, omissions or have suggestions for future improvements, please send us feedback via email feedback@neurotechnology.com.

2 Introduction

Free Fingerprint Verification software development kit (FFV SDK) is a free software component intended for software developers who want to add fingerprint verification functionality in their own software applications. FFV SDK supports various fingerprint scanners and it is able to perform a scanned fingerprint verification against another fingerprint stored in an internal database. The FFV SDK is intended to be used in various logon applications, but it can be also used in any other application.

Additionally, FFV SDK enables developers to use a wide range of programming languages in a development environment of their choice to create applications. This software development kit includes a documentation and sample codes for different programming languages that can be used to guide developers to produce their own applications or add a fingerprint biometric functionality to existing applications.

Free Fingerprint Verification SDK functionality is based on the high level of accuracy Neurotechnology algorithm which is used in VeriFinger SDK and MegaMatcher SDK.

2.1 About This Guide

This document is a developer guide on writing biometrical applications with FFV SDK. When developing your own applications you must be proficient in at least one of these programming languages: C++, C, C#, Delphi, Java, VB .NET, VB6. Also a basic knowledge of biometrical systems is desirable.

2.1.1 How the Guide Is Organized

Chapter *Introduction (see page 2)* focuses on the general information about FFV SDK.

Chapter *Quick Start (see page 14)* provides a quick introduction to the FFV SDK and discusses how to use a fingerprint scanner and sample applications. Also fingerprints enrollment and verification operations are explained.

The components for developing applications that uses the functionality of the FFV SDK are defined in chapter *API Reference (see page 21)*.

Answers to frequently asked questions are reviewed in chapter *FAQ (see page 81)*.

2.1.2 Target Audience

This guide is for developers who have a working experience in programming with at least one of these programming languages: C, C++, C#, Delphi, Java, VB .NET or VB6.

2.2 Additional Resources

This chapter provides additional resources that can help you using the FFV SDK.

2.2.1 About Neurotechnology

Neurotechnology provides algorithms and software development products for biometric fingerprint and face recognition, computer-based vision and object recognition to security companies, system integrators and hardware manufacturers. More than 1,900 system integrators and sensor providers in more than 60 countries license and integrate company's technology into their own products.

Drawing from years of academic research in the fields of neuroinformatics, image processing and pattern recognition, Neurotechnology was founded in 1990 in Vilnius, Lithuania under the name Neurotechnologija and released its first fingerprint identification system in 1991. Since that time Neurotechnology has released more than 40 products and version upgrades for both identification and verification of objects and personal identity.

With a combination of fast algorithms and high reliability, company's fingerprint and face biometric technologies can be used for access control, computer security, banking, time attendance control and law enforcement applications, among others.

Neurotechnology's fingerprint identification algorithm has shown one of the best results for reliability in several biometric competitions, including the International Fingerprint Verification Competition (FVC2006, FVC2004, FVC2002 and FVC2000) and the National Institute of Standards & Technology (NIST) Fingerprint Vendor Technology Evaluation for the US Department of Justice (FpVTE 2003), where Neurotechnology ranked among the top five companies for accuracy in single-finger tests.

2.2.2 Free SDK vs. VeriFinger SDK

Free Fingerprint Verification SDK is based on the same algorithm that is used in VeriFinger SDK and has the same fingerprint features and high matching reliability. VeriFinger SDK is intended for biometric system developers and integrators and allows a rapid development of large-scale biometric applications based on fingerprint verification. Also the VeriFinger SDK can have an additional component - Matching server which performs the identification and verification of fingerprints on server side.

Differences between Free Fingerprint Verification SDK and Verifinger SDK are listed in the table below:

Feature	**FFV SDK**	**VeriFinger SDK**
Fingerprint scanners support	+	+
Fingerprint verification against live scanned image	+	+
High speed identification against database		+
Fingerprint features template extraction from image		+
Programming samples and tutorials	+	+
Database template count*	10	unlimited
Support for Windows operating systems (2000/XP/Vista/7)	+	+
Support for Linux operating systems		+
Support for Mac OS X operating system		+
Support for 64 bit operating systems (Windows and Linux)		+

* Database template count is a maximum number of fingerprints that can be saved to a database.

If you need more information about VeriFinger SDK, please visit http://www.neurotechnology.com/vf_sdk.html.

2.2.3 Online Resources

If you need more information about the company or products you can refer to one of these online resources:

Link	Description
http://www.neurotechnology.com/	The Neurotechnology home page. Provides the latest news and updates of Neurotechnology products.
http://www.neurotechnology.com/neurotec-forum/	The Neurotechnology forum. Provides the peer-to-peer connection between Neurotechnology developers and customers.

2.3 System Requirements

The minimal hardware and software requirements needed to run Neurotechnology Free Fingerprint Verification SDK are listed:

- Microsoft Windows 2000/2003/XP/Vista/7 operating system
- Processor with x86-based architecture
- Fingerprint scanner (for the list of supported scanners see the section Supported Scanners (see page 4))
- A connection for connecting a fingerprint scanner.

2.4 Fingerprint Scanners

The following fingerprint scanners are supported:

Fingerprint scanner model	Description	Requirements
U.are.U 2000S	The U.are.U 2000 fingerprint scanner is a self-contained sensor for capturing a fingerprint and communicating the digital image to PfC via USB interface. The on-board electronics control image capture, self-calibration, and the Plug-n-Play USB interface.	**Drivers**: • \install\Fingerprint Scanners\UareU\ **DLLs**: • `FPSmm\FPSmmUareU.dll`
DigitalPersona U.are.U 4000S / U.are.U 4000B	The U.are.U 4000 fingerprint sensor is designed to work with PC via USB port. It has slim design and small form factor. The on-board electronics control image capture, latent fingerprint rejection, self-calibration, and the Plug-n-Play USB interface.	**Drivers**: • \install\Fingerprint Scanners\UareU\ **DLLs**: • FPSmm\FPSmmUareU.dll
DigitalPersona U.are.U Fingerprint Keyboard	This is 104-key Windows compatible keyboard with a built-in U.are.U 4000 fingerprint sensor. The keyboard requires two connections: PS/2 connection for keyboard functioning and USB for fingerprint scanner.	

DigitalPersona U.are.U 4000 Fingerprint Module	**Description**: The U.are.U 4000 Module is a small fingerprint scanner designed for integration into OEM equipment where fingerprint authentication is needed.	
DigitalPersona U.are.U 4500	DigitalPersona U.are.U 4500 is an optical USB 2.0 fingerprint scanner. The scanner is able to reject latent and spoof fingerprints.	**Drivers**: • \install\Fingerprint Scanners\UareU\ **DLLs:** • FPSmm\FPSmmUareU.dll
Cross Match Verifier 300 Classic	This scanner is intended for professional use. It operates via USB port.	**Drivers**: Can be downloaded from CrossMatch website (section "USB SDK for Verifier and MV5 Scanners/Readers") http://www.crossmatch.com/software.html#_USB_SDK_for **DLLs:** • FPSmm\FPSmmCrossMatch.dll
Cross Match Verifier 300 LC	Verifier 300 LC (Lexan Case) features light weight (less than 0.5 kg). It operates via USB port.	**Drivers**: Can be downloaded from CrossMatch website (section "USB SDK for Verifier and MV5 Scanners/Readers") http://www.crossmatch.com/software.html#_USB_SDK_for **DLLs**: • FPSmm\FPSmmCrossMatch.dll
Cross Match Verifier 300 LC 2.0	An improved version of Verifier 300 LC. Features faster frame rate and an I/R filter to improve ambient light rejection.	**Drivers**: Can be downloaded from CrossMatch website (section "USB SDK for Verifier and MV5 Scanners/Readers") http://www.crossmatch.com/software.html#_USB_SDK_for **DLLs**: • FPSmm\FPSmmCrossMatch.dll
Cross Match Verifier 310	This scanner allows to scan two flat fingerprints simultaneously or one rolled fingerprint.	**Drivers**: Can be downloaded from CrossMatch website (section "USB SDK for Verifier and MV5 Scanners/Readers") http://www.crossmatch.com/software.html#_USB_SDK_for **DLLs**: • FPSmm\FPSmmCrossMatch.dll
Futronic FS80, Futronic FS82	**Remarks**: Configuration `futronic.cfg` file with parameter **LFD = false** will turn of the life fingerprint detection. For BioLink U-Match MatchBook v.3.5 scanner file with parameter **LFD = false** should be created all the time.	**Drivers**: • \install\Fingerprint Scanners\Futronic\ **DLLs**: • FPSmm\FPSmmFutronic.dll • FPSmm\ftrScanAPI.dll

Futronic eFAM (FS84)	Futronic FS84 is an ethernet Fingerprint Authentication Module (eFAM) suitable for embedded, regular or remote applications. The module can be controlled via standard Ethernet interface by any host system or via serial interface.
	1 output channel (for external relay control) and 2 input channels (for door sensor and switch) are available for external device control. Electric lock or other electric device can be activated by eFAM using these output control signals.
	Remarks: `FPSmmFutronicEthernetFam.ini` file is intended for scanner configuration. Scanner IP address and port should be placed in file. Examples:
	`192.168.2.255 5001`
	or for few scanners
	`192.168.2.253 5001`
	`192.168.2.254 5001`
	`192.168.2.255 5005`
		Drivers:
		• \install\Fingerprint Scanners\FutronicEthernetFam\
		DLLs:
		• FPSmm\FPSmmFutronicEthernetFam.dll

Futronic FS88	The scanner is an enhanced version of Futronic FS80 scanner. This scanner was certified by FBI to be compliant with PIV-071006 Image Quality Specification for Singer Finger Reader. The FS88 scanner includes an electronic circuit for **live finger detection**. **Remarks**: Configuration `futronic.cfg` file with parameter **LFD = false** will turn of the life fingerprint detection. For BioLink U-Match MatchBook v.3.5 scanner file with parameter **LFD = false** should be created all the time.	**Drivers**: • \install\Fingerprint Scanners\Futronic\ **DLLs**: • FPSmm\FPSmmFutronic.dll • `FPSmm\ftrScanAPI.dll`
Futronic FS50	This scanner can capture single finger, dual finger and roll finger image. The scanner can also handle bad fingers such as dry, wet, blurred and scarred without any problem. **Remarks**: Configuration `futronic.cfg` file with parameter **LFD = false** will turn of the life fingerprint detection.	**Drivers**: • \install\Fingerprint Scanners\Futronic\ **DLLs**: • FPSmm\FPSmmFutronic.dll • `FPSmm\ftrScanAPI.dll`
Futronic FS60	FS60 is professional fingerprint scanner which can capture high quality 4 fingerprints image.	**Drivers**: • \install\Fingerprint Scanners\FutronicMF\ **DLLs**: • FPSmm\FPSmmFutronicMF.dll • `FPSmm\ftrMFAPI.dll`
NITGEN eNBioScan-F	The scanner meets FBI's Integrated AFIS Image Quality Specifications (IQS) and is able to scan wet fingers. **Remarks**: `FPSmmNitgen.dll` from *Additional* folder should be copied to *FPSmm* folder in order to use this scanner driver.	**Drivers**: • \install\Fingerprint Scanners\Nitgen\ **DLLs**: • FPSmm\Additional\FPSmmNitgen.dll • FPSmm\NBioBSP.dll

NITGEN Fingkey Hamster	**Remarks**: `FPSmmNitgen.dll` from *Additional* folder should be copied to *FPSmm* folder in order to use this scanner driver.	**Drivers**: • \install\Fingerprint Scanners\Nitgen\ **DLLs**: • FPSmm\Additional\FPSmmNitgen.dll • FPSmm\NBioBSP.dll
NITGEN Fingkey Hamster II	**Remarks**: `FPSmmNitgen.dll` from *Additional* folder should be copied to *FPSmm* folder in order to use this scanner driver.	**Drivers**: • \install\Fingerprint Scanners\Nitgen\ **DLLs**: • FPSmm\Additional\FPSmmNitgen.dll • FPSmm\NBioBSP.dll
SecuGen Hamster III scanner	**Remarks**: `FPSmmSecugenHFDU03.dll` from *Additional* folder should be copied to *FPSmm* folder in order to use this scanner driver.	**Drivers**: • \install\Fingerprint Scanners\SecuGenHFDU03\ **DLLs**: • FPSmm\Additional\FPSmmSecugenHFDU03.dll
SecuGen Hamster Plus scanner	**Remarks**: `FPSmmSecugenHFDU02.dll` from *Additional* folder should be copied to *FPSmm* folder in order to use this scanner driver.	**Drivers**: • \install\Fingerprint Scanners\SecuGenHFDU02\ **DLLs**: • fpsmm\Additional\FPSmmSecugenHFDU02.dll
SecuGen Hamster IV scanner	**Remarks**: `FPSmmSecugenHFDU04.dll` from *Additional* folder should be copied to *FPSmm* folder in order to use this scanner driver.	**Drivers**: • \install\Fingerprint Scanners\SecuGenHFDU04\ **DLLs**: • fpsmm\Additional\FPSmmSecugenHFDU04.dll
Dermalog ZF1	The scanner is able to detect fake fingers and to scan both dry and wet fingerprints.	**Drivers**: • \install\Fingerprint Scanners\DermalogZF1\ **DLLs**: • FPSmm\Additional\FPSmmDermalogZF1.dll • FPSmm\DermalogVC3.dll • FPSmm\DermalogCalibrateSDK.dll • FPSmm\DermalogLoggingFacility.dll • FPSmm\DermalogPLS1.dll • FPSmm\ZFScanAPI.dll
BioLink U-Match MatchBook v.3.5		**Drivers**: • \install\Fingerprint Scanners\Futronic\ **DLLs**: • FPSmm\FPSmmFutronic.dll • FPSmm\ftrScanAPI.dll

Testech Bio-i		**Drivers**: • \install\Fingerprint Scanners\Testech\ **DLLs**: • FPSmm\FPSmmCyte.dll • FPSmm\BioNetCapture.dll
Startek FM200 scanner		**Drivers**: • \install\Fingerprint Scanners\Startek\ **DLLs**: • FPSmm\FPSmmFM200.dll • FPSmm\fm200api.dll • FPSmm\FingerPrinterDll.dll • FPSmm\fm200drv.dll
Tacoma CMOS Scanner		**Drivers**: • \install\Fingerprint Scanners\Tacoma\ **DLLs**: • FPSmm\FPSmmTacoma.dll • FPSmm\SmzCmos1.dll • FPSmm\SMZ_API.dll
Fujitsu MBF200		**Drivers**: • \install\Fingerprint Scanners\Fujitsu\ **DLLs**: • FPSmm\FPSmmFujitsu.dll • FPSmm\libusb0.dll
Identix DFR 2080	**Remarks**: Drivers for this scanner are not included into the Free Fingerprint Verification SDK. **Remarks**: `FPSmmIdentixR.dll` from *Additional* folder should be copied to *FPSmm* folder in order to use this scanner driver.	**DLLs**: • FPSmm\Additional\FPSmmIdentixR.dll • FPSmm\Itf32_2080U2.dll
Identix DFR 2090	This scanner is intended for professional use. The image output is in USB digital and RS-170 analog video formats. **Remarks**: Drivers for this scanner are not included into the Free Fingerprint Verification SDK.	**DLLs**: • FPSmm\Additional\FPSmmIdentixR.dll • FPSmm\Itf32_2080U2.dll
Identix DFR 2100		**DLLs**: • FPSmm\Additional\FPSmmIdentixR.dll • FPSmm\Itf32_2080U2.dll
TST Biometrics BiRD 3	TST Biometrics offers its *touchless sensor technology* that allows to scan a finger without physical contact with a fingerprint sensor. The BiRD 3 sensor is available as desktop scanner, on-wall mounted scanner and as OEM components. Optionally, a 5V AC powered heating device could be included for operating in cold environment.	**Drivers** can be downloaded from TST Biometrics website http://www.tst-biometrics.com/. **DLLs**: • FPSmm\Additional\FPSmmTSTBIRD.dll • FPSmm\TSTBasic.dll

Digent Izzix FD1000		**Drivers**: • \install\Fingerprint Scanners\Digent\ **DLLs**: • FPSmm\FPSmmDigent.dll • FPSmm\IZZIX.dll
UPEK TouchChip TCRU1C	This scanner is built on the TouchChip Silicon Fingerprint Sensor. It communicates PC via USB port.	**Drivers**: • \install\Fingerprint Scanners\Upek\ **DLLs**: • FPSmm\FPSmmUpek.dll • FPSmm\TCI.dll
UPEK TouchChip TCRU2C		**Drivers**: • \install\Fingerprint Scanners\Upek\ **DLLs**: • FPSmm\FPSmmUpek.dll • FPSmm\TCI.dll
UPEK Eikon		**Drivers**: • \install\Fingerprint Scanners\Upek\ **DLLs**: • FPSmm\FPSmmEikon.dll
Eikon Touch 700, Eikon Touch 300		**Drivers**: • \install\Fingerprint Scanners\Upek\ **DLLs**: • FPSmm\iaapi.dll • FPSmm\FPSmmUpek.dll
Green Bit DactyScan 26	**Remarks**: Drivers for this scanner are not included into the Free Fingerprint Verification SDK.	**DLLs**: • FPSmm\FPSmmDactyScan.dll • FPSmm\FSM26U.dll
Hongda S680	This scanner allows to scan rolled fingerprints. A plastic lid can be mounted on top of sensor for more comfortable flat fingerprint scanning.	**Drivers**: • \install\Fingerprint Scanners\Hongda\ **DLLs**: • FPSmm\FPSmmHongda.dll
Jstac Athena 210		**Drivers**: • \install\Fingerprint Scanners\Jsatck\ **DLLs**: • FPSmm\FPSmmJstac.dll • FPSmm\WIS_API.dll • FPSmm\WisCmos2.dll

BiometriKa FX 2000	BiometriKa FX 2000 desktop fingerprint scanner is intended for using with PC. Scanner communicates PC via USB interface. FX 2000 contains 32 bit RISC processor for encrypting fingerprint data, controlling scanner operations and other operations.	**Drivers**: • \install\Fingerprint Scanners\BiometriKa\ **DLLs**: • FPSmm\FPSmmBiometrika.dll • FPSmm\fx3.dll • FPSmm\fx3scan.dll
BiometriKa FX 3000		**Drivers**: • \install\Fingerprint Scanners\BiometriKa\ **DLLs**: • FPSmm\FPSmmBiometrika.dll • FPSmm\fx3.dll • FPSmm\fx3scan.dll
BiometriKa HiScan		**Drivers**: • \install\Fingerprint Scanners\BiometriKa\ **DLLs**: • FPSmm\FPSmmBiometrika.dll • FPSmm\fx3.dll • FPSmm\fx3scan.dll
Lumidigm Venus Series sensors	**Remarks**: Configuration `lumidigm.cfg` file with parameter **LFD = true** will turn on the life fingerprint detection, parameter **LFDThreshold=7000** is life fingerprint detection threshold, the **LFDThreshold** parameter rage is [0-4294967296). `lumidigm.cfg` file should be placed near `FPSmmLumidigm.dll` file. Default parameters are **LFD = false** and **LFDThreshold=7000**	**Drivers**: • \install\Fingerprint Scanners\Lumidigm\ **DLLs**: • FPSmm\LumiAPI.dll • FPSmm\LumiCore.dll • FPSmm\FPSmmLumidigm.dll
Dakty Fingerprint NAOS-A	A fiber optic fingerprint sensor with live finger detection using human body capacitance, blood oxygen presence and pulse measuring.	**Drivers**: • \install\Fingerprint Scanners\DaktyFpd\ **DLLs**: • FPSmm\FPSmmDaktyScan.dll • FPSmm\DaktyImage.dll • FPSmm\fpd.dll • FPSmm\fpdusb.dll • FPSmm\Segmentation.dll
id3 Certis Image	An Atmel FingerChip based scanner with a sweeping thermal sensor.	**Drivers**: • \install\Fingerprint Scanners\Certis\ **DLLs**: • FPSmm\FPSmmCertis.dll • FPSmm\CertisExports.dll • FPSmm\Id3BiokeyDll.dll

CS-Pass USB Fingerprint Sensor	The CS-Pass USB Fingerprint Sensor is based on AuthenTec AES4000 sensor. It is suitable for PC and mobile devices, including battery powered devices. The sensor can be customized for specific projects.	
EntréPad AES2501B		
EntréPad AES4000	The AES4000 fingerprint sensor is suitable for PC and mobile devices. Sensor's small size and low power is ideally suited for battery powered devices.	
FingerLoc AF-S2	The AF-S2 fingerprint sensor is suitable for the embedded devices.	
LTT-C500 Fingerprint Sensor		**DLLs**: • FPSmm\FPSmmLighTunning.dll • FPSmm\GetImageC500.dll
Atmel FingerChip		**Drivers**: • \install\Fingerprint Scanners\AtmelFC\ **DLLs**: • FPSmm\FPSmmAtmel.dll • FPSmm\FingerChip.dll
Zvetco Verifi P4000	An USB 2.0 scanner based on AES4000 capacitive sensor.	**Drivers**: • \install\Fingerprint Scanners\AuthenTec **DLLs**: • FPSmm\Additional\FPSmmAuthentec2501.dll
Zvetco Verifi P5000	A **FIPS-201 compliant** USB 2.0 fingerprint scanner. The scanner is based on the UPEK TCR1 capacitive sensor, that is also used in TCRU1C fingerprint scanner. P5000 scanner is rugged and scratch resistant. Scanner's sensor has protective coating that is able to withstand more than 10 million touches.	**Drivers**: • \install\Fingerprint Scanners\Upek **DLLs**: • FPSmm\FPSmmUpek.dll • FPSmm\TCI.dll
ZKSoftware ZK6000	ZK6000 is an optical USB 2.0 fingerprint scanner. The scanner is able to reject latent and spoof fingerprints.	**Drivers**: • \install\Fingerprint Scanners\ZKSensor6000\ **DLLs**: • FPSmm\FPSmmZKSensor6000.dll
CrossMatch L Scan Guardian	CroosMatch L Scan Guardian is live ten-fingerprints scanner. **Remarks**: Before starting the scanner its sensor should be cleaned, otherwise it will not be able to initialize. For 4 fingerprints scanning fingerprints segmentation module is required. This module is included only in MegaMatcher SDK.	**Drivers:** • \install\Fingerprint Scanners\LScanEssentials\ **DLLs**: • FPSmm\FPSmmCrossMatch.dll

Suprema BioMini	Suprema BioMini is a high performance USB fingerprint scanner.	**Drivers**: • \install\Fingerprint Scanners\Suprema\ **DLLs**: • FPSmm\FPSmmSuprema.dll
Suprema SFR300-S	Suprema SFR300-S is a high performance USB fingerprint scanner for fingerprint authentication in desktop or network security.	**Drivers**: • \install\Fingerprint Scanners\Suprema\ **DLLs**: • FPSmm\FPSmmSuprema.dll
Suprema SFR300-R		**Drivers**: • \install\Fingerprint Scanners\SupremaSFR300R\ **DLLs**: • FPSmm\FPSmmSupremaSFR300R.dll
Suprema RealScan-D		**Drivers**: • \install\Fingerprint Scanners\Suprema\ **DLLs**: • FPSmm\FPSmmRealScan.dll **Note**: `realscan.cfg` configuration file should be created in FPSmm folder. When mode value is `rolled` one fingerprint can be scanned. When the value is `flat`, one, two or four fingerprints can be scanned.
Suprema RealScan-10		**Drivers**: • \install\Fingerprint Scanners\Suprema\ **DLLs**: • FPSmm\FPSmmRealScan.dll **Note**: `realscan.cfg` configuration file should be created in FPSmm folder. When mode value is `rolled` one fingerprint can be scanned. When the value is `flat`, one, two or four fingerprints can be scanned.
Suprema SFR3000		**Drivers**: • \install\Fingerprint Scanners\Suprema\ **DLLs**: • FPSmm\FPSmmRealScan.dll
Futronic FS90		**Drivers**: • \install\Fingerprint Scanners\Futronic\ **DLLs**: • FPSmm\FPSmmFutronic.dll • `FPSmm\ftrScanAPI.dll`

SOP1		**Drivers**: • \install\Fingerprint Scanners\SOP1\ **DLLs**: • FPSmm\FPSmmSOP1.dll • FPSmm\sop1.dll
Vista MT		**Drivers:** • \install\Fingerprint Scanners\VistaMT\ **DLL:** • FPSmm\FPSmmVistaMT.dll • FPSmm\VciMt.dll
LES Fingerprint scanner from Integrated Biometrics		**Drivers:** • \install\Fingerprint Scanners\Cyte\ **DLL:** • FPSmm\FPSmmCyte.dll • FPSmm\BioNetCapture.dll
Nitgen Fingkey Mouse I, II, III	**Remarks**: `FPSmmNitgen.dll` from *Additional* folder should be copied to FPSmm folder in order to use this scanner driver.	**Drivers**: • \install\Fingerprint Scanners\NITGEN\ **DLLs**: • fpsmm\Additional\FPSmmNitgen.dll • fpsmm\NBioBSP.dll

3 Quick Start

This chapter provides a quick introduction to FFV SDK. In this chapter is discussed:

- The basic terminology related to fingerprints.
- The instructions for using a fingerprint scanner.
- A guide for using sample applications which are included in FFV SDK.

3.1 Terminology

The following two main definitions which are the cornerstone of biometric systems that use fingerprint identification will help you understand the Neurotechnology Free Fingerprint Verification SDK functionality:

- **Enrollment (see page 14)** is the process of capturing a person's fingerprint (using a fingerprint capture device). During the enrollment process the FFV SDK saves a person's fingerprint to a database.
- **Verification (see page 15)** is the process when a captured fingerprint is compared to an enrolled fingerprint in order to determine whether the two match.

Fingerprint enrollment and verification processes are described in detail in chapter Fingerprints (see page 14).

3.2 Fingerprints

A **fingerprint** is an impression of the friction ridges of all or any part of the finger. Fingerprint recognition systems use characteristics from these ridges (they are also called fingerprint features) to differentiate one fingerprint from another. The Free Fingerprint Verification SDK converts these features to a format (a template) that enables to perform fingerprint enrollment and verification operations efficiently and qualitatively.

3.2.1 Enrollment

Fingerprint enrollment is a process during which features from a finger are extracted and saved as a fingerprint template for a future comparison against other fingerprint templates. The following instructions describe a typical fingerprint enrollment scheme (the same scheme is used in sample applications):

1. Get a person's identification number.
2. Capture a person's fingerprint using a fingerprint scanner.
3. Extract a fingerprint features from a fingerprint image.
4. Associate a person with his fingerprint.
5. Save extracted features (a template) to a database.

To enroll a fingerprint you can use these functions:

```
//C/C++ function
NResult N_API NffvEnroll(HNffv hFfv, NUInt timeout, NffvStatus * pStatus, HNffvUser *
pHUser);
```

```
//.NET method
public NffvUser Enroll(uint timeout, out NffvStatus status);
```

3.2.2 Verification

Fingerprint verification is a process during which a scanned fingerprint is compared with the one saved to a database and is decided whether the two match. The following scheme is usually used for a fingerprint verification:

1. Get a person's identification number.
2. Capture a person's fingerprint using a fingerprint scanner.
3. Extract a fingerprint features from a fingerprint image for the purpose of verification.
4. Get a fingerprint template (the one that was saved to a database earlier) by identification number
5. Compare two fingerprints: the one that was scanned with the one that was saved to database.
6. Perform an action according to the verification result (eg. unlock a computer if two fingerprints matches).

To verify a fingerprint you can use these functions:

```
//C/C++ function
NResult N_API NffvVerify(HNffv hFfv, HNffvUser hUser, NUInt timeout, NffvStatus * pStatus,
NInt * pScore);
//.NET method
public int Verify(NffvUser user, uint timeout, out NffvStatus status);
```

3.3 Quality Threshold

Quality threshold is the property that defines a scanned fingerprint image quality. Specifies the threshold which is considered when extracting fingerprint features from the image. With higher threshold better quality of fingerprint image is required to successfully extract fingerprint features. The Quality threshold should be in range [0, 255]. 255 means the best image quality.

The default quality threshold value is 100.

In the FFV SDK quality threshold can be set using these functions:

```
//.NET property
public byte QualityThreshold;
//C/C++ function
NResult N_API NffvSetQualityThreshold(HNffv hEngine, NByte value);
```

3.4 Matching Threshold

Matching threshold - the minimum similarity value that verification and identification functions accept for the same finger fingerprints or face.

Matching threshold should be selected according to the system's development requirements. The default value is 48.

In the FFV SDK matching threshold can be set using these functions and properties:

```
//.NET property
public int MatchingThreshold;
//C/C++ function
NResult N_API NffvSetMatchingThreshold(HNffv hEngine, NInt value);
```

3.5 How to Use Fingerprint Scanner

A **fingerprint scanner** is a device connected to computer and used for capturing a person's fingerprint image. Depending on scanner's manufacturer and model it can be connected to USB or Ethernet port. In order to use a fingerprint scanner with the FFV SDK you should do the following:

- Insert a fingerprint scanner into the USB or Ethernet connector on the system where you copied the FFV SDK files.
- Install the scanner drivers whether the one you got from a manufacturer (yours) or from the FFV SDK folder (*\install\Fingerprint Scanners*).

Notes

It is recommended to use drivers from *\install\Fingerprint Scanners* folder.

3.6 Using Sample Applications

The FFV SDK contains sample applications which demonstrates the functionality of the FFV SDK. You are free to adjust these applications for your needs.

The sample applications are located in "*\Samples*" folder. If you want to test the sample application from this folder, you must first compile or build files from this folder.

There are samples in the following programming languages:

- C++ (*\Samples\Cpp*)
- C# (*\Samples\CSharp*)
- VB .NET (*\Samples\VB.NET*)
- Java (*\Samples\Java*)
- Delphi (*\Samples\Delphi*)
- VB6 (*\Samples\VB6*)

This chapter explains in detail how to use C# sample applications and comments the source code. The usage of samples in different programming languages are very similar.

By reading this section you will

- Open a sample application project file and build it
- Enroll a fingerprint
- Make a verification of a fingerprint

If you want to test a sample application without building it, you can find an executable file in *\bin\Win32_x86*.

Using C# sample application

1. **Starting the sample aplication**

Open the solution file using Microsoft Visual Studio 2005 located in the folder "*\Samples\CSharp\CSharpSample.sln*".

The C# sample solution project contains these main files:

- *AboutForm.cs*. This file is used for showing a basic information about a sample application.
- *ChooseScannerForm.cs*. This file is used for showing a dialog box for selecting a fingerprint scanner.
- *SettingsForm.cs*. This file is used for showing a form where matching and quality thresholds can be set.

- *MainForm.cs*. This file contains all the main functionality of the application (also methods for fingerprint enrollment and verification).
- *UserInfoForm.cs*. This file contains properties that enable to get or set a user name and fingerprint.

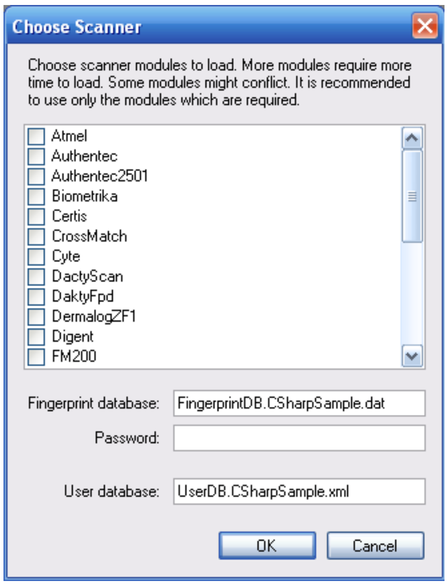
Also you should notice that the solution project contains references to these dynamic-link libraries (DLL):

- **Neurotec.dll**. This library provides an internal functionality for the FFV SDK. You don't need to use methods from this library, but it should be included into your solution project.
- **Neurotec.Biometrics.Nffv.dll**. This library is the main DLL for your solution projects and provides the enrollment and verification of a fingerprint functionality.

These libraries can be located in the "*\bin\Win32_x86*" folder of the FFV SDK.

2. **Selecting a fingerprint scanner**

When you have built the sample application solution project and launched it, the dialog box for selecting a scanner appears:


Choose Scanner

Choose scanner modules to load. More modules require more time to load. Some modules might conflict. It is recommended to use only the modules which are required.

- [ ] Atmel
- [ ] Authentec
- [ ] Authentec2501
- [ ] Biometrika
- [ ] Certis
- [ ] CrossMatch
- [ ] Cyte
- [ ] DactyScan
- [ ] DaktyFpd
- [ ] DermalogZF1
- [ ] Digent
- [ ] FM200

Fingerprint database: FingerprintDB.CSharpSample.dat

Password:

User database: UserDB.CSharpSample.xml

There are listed scanner models supported by the FFV SDK. Select only the scanner models you will use. You should note that more modules require more time to load.

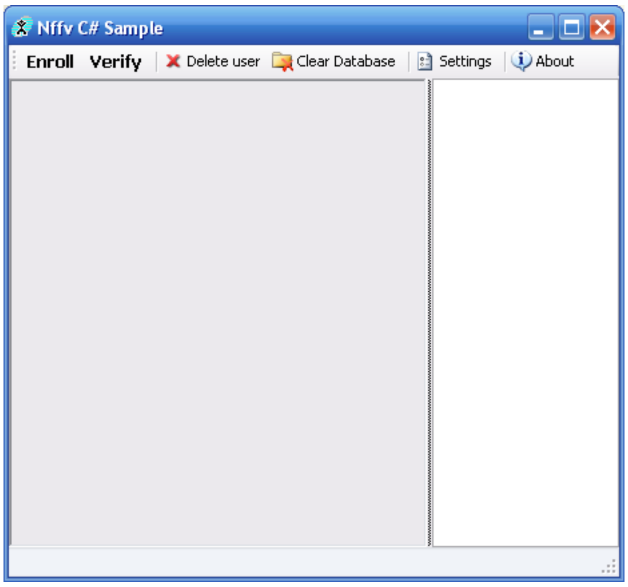
Enrolled fingerprints will be saved to a database (see a fingerprint database field). You can protect this database by setting a password. Person's details are saved to users database (in this sample application users database is an Xml file) where a person's name and his fingerprint ID is saved. You can implement your own users database by adding more fields.

To get the list of scanners supported by the FFV SDK, you can use the following code:

```
//A string variable which contains a list of scanner modules
private string scannerModules = string.Empty;

//Gets a list of supported scanners
scannerModules = NFEngine.GetAvailableScannerModules();
```

3. **The main window of the sample application**

After you have selected fingerprint scanners, created databases and pressed the OK button, the main window of the sample application appears:



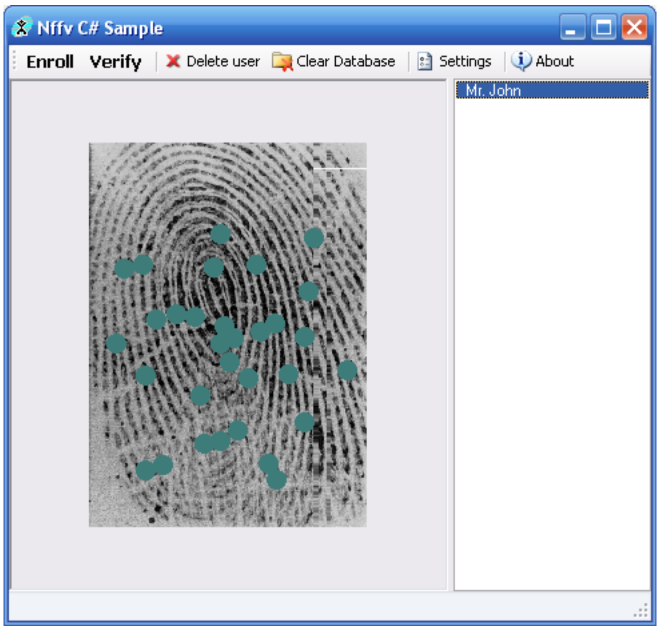
Below are listed operation you can do using this sample application:

- Enroll
- Verify
- Delete user
- Clear database
- Settings
- About

Now let's discuss these operations in detail and illustrate them using C# source code.

4. **Enrolling a fingerprint**

With the purpose of enrolling a fingerprint to database a fingerprint scanner should be connected to a computer. The fingerprint is enrolled by pressing "Enroll" (the dialog box shows up and asks for a name of a person). After the enrollment of a person's fingerprint has finished you should see a window like this:



A fingerprint image and the name of a person (let's say it is Mr. John) is shown on a window.

Using The FFV SDK you can enroll up to 10 records to a database.

Person's fingerprint can be enrolled using this C# method:

```
public void EnrollUser ( )
{
    RunWorkerCompletedEventArgs taskResult = BusyForm.RunLongTask("Waiting for fingerprint
...",
        new DoWorkEventHandler(doEnroll),
        false, null, new EventHandler(CancelScanningHandler));
    EnrollmentResult enrollmentResult = (EnrollmentResult)taskResult.Result;
    if (enrollmentResult.engineStatus == NffvStatus.TemplateCreated)
    {
         NffvUser engineUser = enrollmentResult.engineUser;
         string userName = enrollDlg.UserName;
         if (userName.Length <= 0)
             {
                 userName = engineUser.Id.ToString();
              }

          _userDB.Add(new UserRecord(engineUser.Id, userName));
          try
          {
              _userDB.WriteToFile(_userDatabaseFile);
           }
          catch { }

          pbExtractedImage.Image = engineUser.GetBitmap();
          lbDatabase.Items.Add(new ListBoxImage.CData(engineUser, userName));
          lbDatabase.SelectedIndex = lbDatabase.Items.Count - 1;
    }
    else
    {
        NffvStatus engineStatus = enrollmentResult.engineStatus;
        MessageBox.Show(string.Format("Enrollment was not finished. Reason: {0}",
engineStatus));
     }
}
```

5. **Verifying a fingerprint**

When you need to verify a person's fingerprint with the one that was enrolled to a database you should select a database record and press the "Verify" button. After your fingerprint is scanned the verification is made. If the two fingerprints are identical, the matching score is shown. Otherwise, a message box announcing that fingerprints are not identical is shown.

Person's scanned fingerprint verification can be made using this C# method:

```
private void doVerify(object sender, DoWorkEventArgs args)
{
    VerificationResult verificationResult = new VerificationResult();
    verificationResult.score = _engine.Verify((NffvUser)args.Argument, 20000,
                                 out verificationResult.engineStatus);
    args.Result = verificationResult;
}
```

6. **Deleting a user**

To delete a user from a database you can use this C# method:

```
private Neurotec.Biometrics.Gui.ListBoxImage lbDatabase;
UserDatabase _userDB;
Nffv _engine;

public void DeleteUser ( )
{
   if (lbDatabase.SelectedIndex < 0)
   {
      MessageBox.Show("Please select a record from the database.");
   }
   else
   {
      _userDB.Remove(_userDB.Lookup(((ListBoxImage.CData)lbDatabase.SelectedItem).ID));
      try
      {
```

3

```
            _userDB.WriteToFile(_userDatabaseFile);
        }
        catch { }

        _engine.Users.RemoveAt(lbDatabase.SelectedIndex);
        lbDatabase.Items.RemoveAt(lbDatabase.SelectedIndex);
    }
}
```

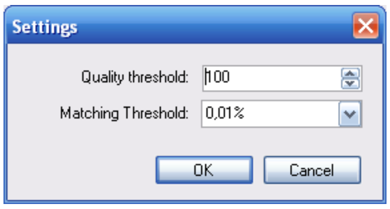
7. **Clearing a database**

When you need to delete all records from a database you can use this C# method:

```
//Fields
Nffv _engine;
private Neurotec.Biometrics.Gui.ListBoxImage lbDatabase;
UserDatabase _userDB;

public void ClearDatabase ( )
{
   if (MessageBox.Show("All records will be deleted from database. Do you want to
continue?",
                       "Confirm delete", MessageBoxButtons.YesNo,
                        MessageBoxIcon.Question) != DialogResult.Yes)
   {
      return;
   }

   //Clears a database
   _engine.Users.Clear();
   lbDatabase.Items.Clear();

   _userDB.Clear();
   try
   {
      _userDB.WriteToFile(_userDatabaseFile);
   }
   catch { }
}
```

8. **Settings**

When you press the "Settings" button you will see a window like this:



For more information on setting quality and matching thresholds see chapters Quality Threshold (see page 15) and Matching Threshold (see page 15).

4 API Reference

This chapter defines the components for developing applications that uses the functionality of the VeriFinger Free SDK.

Modules

Name	Description
C/C++ Reference (↗ see page 21)	This chapter provides the Free Fingerprint Verification SDK programming reference for C/C++ programming languages.
.NET Reference (↗ see page 32)	This chapter provides the Free Fingerprint Verification SDK programming reference for Microsoft .NET framework.
Java Reference (↗ see page 41)	This chapter provides the Free Fingerprint Verification SDK programming reference for Java programming language.
Delphi Reference (↗ see page 61)	This chapter provides the Free Fingerprint Verification SDK programming reference for Delphi programming language.

4.1 C/C++ Reference

This chapter provides the Free Fingerprint Verification SDK programming reference for C/C++ programming languages.

Remarks

If you are developing your own application using C or C++, you should link Nffv.dll.lib library to your solution project. Also Nffv.dll library is needed.

In order to use Nffv.dll the folder must contain NffvServer.exe file.

Functions

	Name	Description
◆	NffvCancel (↗ see page 23)	Cancels a fingerprint enrollment or verification operation.
◆	NffvClearUsers (↗ see page 23)	Removes all the users which were enrolled to a database.
◆	NffvEnroll (↗ see page 24)	Gets a fingerprint from a scanner and saves it to a database.
◆	NffvFreeMemory (↗ see page 24)	Releases memory allocated by the NffvGetAvailableScannerModules function..
◆	NffvGetAvailableScannerModulesA (↗ see page 24)	Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.
◆	NffvGetAvailableScannerModulesW (↗ see page 25)	Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.
◆	NffvGetErrorMessageA (↗ see page 25)	Gets an error message. Use this function for errors handling.
◆	NffvGetErrorMessageW (↗ see page 25)	Gets an error message. Use this function for errors handling.
◆	NffvGetMatchingThreshold (↗ see page 26)	Gets the minimum similarity value that verification function uses to determine whether the fingerprint matches.
◆	NffvGetQualityThreshold (↗ see page 26)	Gets an image quality threshold.
◆	NffvGetUser (↗ see page 26)	Gets the information from a users list about an enrolled user.
◆	NffvGetUserById (↗ see page 27)	Returns user details by the Id from a database.
◆	NffvGetUserCount (↗ see page 27)	Retrieves the number of users enrolled to database.
◆	NffvGetUserIndexById (↗ see page 27)	Retrieves the index from users list of a user indicated by the Id.

◆	NffvInitializeA (see page 28)	Initializes the FFV. This function as a parameters takes a name and a password of a previously created database or creates a new database.
◆	NffvInitializeW (see page 28)	Initializes the FFV. This function as a parameters takes a name and a password of a previously created database or creates a new database.
◆	NffvRemoveUser (see page 29)	Removes a user from users list (database).
◆	NffvSetMatchingThreshold (see page 29)	Sets the minimum similarity value that verification function uses to determine whether the fingerprint matches.
◆	NffvSetQualityThreshold (see page 29)	Sets an image quality threshold.
◆	NffvUninitialize (see page 30)	Releases memory resources.
◆	NffvUserGetHBitmap (see page 30)	Gets a handle to the bitmap of a user fingerprint.
◆	NffvUserGetImage (see page 30)	Gets a user's fingerprint image which was enrolled to a database.
◆	NffvVerify (see page 31)	Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

Macros

Name	Description
NFFV_MAX_USER_COUNT (see page 32)	The maximum number of users that can be enrolled to a database.

Types

Name	Description
NffvStatus (see page 32)	Enumerates enrollment or verification values of the Nffv (see page 61).

4.1.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
◆	NffvCancel (see page 23)	Cancels a fingerprint enrollment or verification operation.
◆	NffvClearUsers (see page 23)	Removes all the users which were enrolled to a database.
◆	NffvEnroll (see page 24)	Gets a fingerprint from a scanner and saves it to a database.
◆	NffvFreeMemory (see page 24)	Releases memory allocated by the NffvGetAvailableScannerModules function..
◆	NffvGetAvailableScannerModulesA (see page 24)	Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.
◆	NffvGetAvailableScannerModulesW (see page 25)	Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.
◆	NffvGetErrorMessageA (see page 25)	Gets an error message. Use this function for errors handling.
◆	NffvGetErrorMessageW (see page 25)	Gets an error message. Use this function for errors handling.
◆	NffvGetMatchingThreshold (see page 26)	Gets the minimum similarity value that verification function uses to determine whether the fingerprint matches.
◆	NffvGetQualityThreshold (see page 26)	Gets an image quality threshold.
◆	NffvGetUser (see page 26)	Gets the information from a users list about an enrolled user.
◆	NffvGetUserById (see page 27)	Returns user details by the Id from a database.
◆	NffvGetUserCount (see page 27)	Retrieves the number of users enrolled to database.
◆	NffvGetUserIndexById (see page 27)	Retrieves the index from users list of a user indicated by the Id.

	NffvInitializeA (see page 28)	Initializes the FFV. This function as a parameters takes a name and a password of a previously created database or creates a new database.
	NffvInitializeW (see page 28)	Initializes the FFV. This function as a parameters takes a name and a password of a previously created database or creates a new database.
	NffvRemoveUser (see page 29)	Removes a user from users list (database).
	NffvSetMatchingThreshold (see page 29)	Sets the minimum similarity value that verification function uses to determine whether the fingerprint matches.
	NffvSetQualityThreshold (see page 29)	Sets an image quality threshold.
	NffvUninitialize (see page 30)	Releases memory resources.
	NffvUserGetHBitmap (see page 30)	Gets a handle to the bitmap of a user fingerprint.
	NffvUserGetImage (see page 30)	Gets a user's fingerprint image which was enrolled to a database.
	NffvVerify (see page 31)	Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

Module

C/C++ Reference (see page 21)

4.1.1.1 NffvCancel Function

Cancels a fingerprint enrollment or verification operation.

C++

```
NResult N_API NffvCancel();
```

Returns

If the function succeeds the return value is N_OK. Otherwise the error code (see page 80) is returned.

Remarks

This method is useful when the fingerprint enrollment or verification operation take too long. In this case a cancel dialog can be shown for a user to cancel this operation.

Example

This C++ example demonstrates how to stop an enrollment and verification operation:

```
//...
//Function for cancelling enrollment and verification
void OnCancelScan()
{
    NffvCancel();
}
```

4.1.1.2 NffvClearUsers Function

Removes all the users which were enrolled to a database.

C++

```
NResult N_API NffvClearUsers();
```

Returns

If the function succeeds, the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

This functions removes all the users that were enrolled to a database, so be careful when using this function.

4.1.1.3 NffvEnroll Function

Gets a fingerprint from a scanner and saves it to a database.

C++

```
NResult N_API NffvEnroll(NUInt timeout, NffvStatus * pStatus, HNffvUser * pHUser);
```

Parameters

Parameters	Description
NUInt timeout	[in] Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than *timeout* milliseconds.
NffvStatus * pStatus	[out] Enrollment (see page 14) status value enumerated by the NffvStatus (see page 32) enumeration.
HNffvUser * pHUser	[out] A pointer to the FFV user object that provides functions for managing enrolled users.

Returns

If the function succeeds the N_OK value is returned. Otherwise, an error code (see page 80) is returned.

Example

This C++ code demonstrates how to enroll a user:

```
UINT CMainForm::EnrollUserThread(LPVOID pParam)
{
    EnrollParam *pEnrollParam = (EnrollParam*)pParam;
    CMainForm* form = (CMainForm*)pEnrollParam->pMainForm;

    NResult result = NffvEnroll(form->m_iEnrollTimeout, &pEnrollParam->engineStatus,
                                pEnrollParam->pHUser);
    if(NFailed(result)) throw result;

    pEnrollParam->pBusyForm->Stop();

    return 0;
}
```

4.1.1.4 NffvFreeMemory Function

Releases memory allocated by the NffvGetAvailableScannerModules function..

C++

```
void N_API NffvFreeMemory(void * pBlock);
```

Parameters

Parameters	Description
void * pBlock	[out] A pointer to memory block that should be released.

Returns

If the functions succeeds the return value is N_OK. Otherwise, the function returns an error code (see page 80).

4.1.1.5 NffvGetAvailableScannerModulesA Function

Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.

C++

```
NResult N_API NffvGetAvailableScannerModulesA(NAChar * * pSzValue);
```

Parameters

Parameters	Description
NAChar * * pSzValue	[out] A string that contains the list of scanners separated by semicolons.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

This function is an ANSI version of the function.

4.1.1.6 NffvGetAvailableScannerModulesW Function

Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.

C++

```
NResult N_API NffvGetAvailableScannerModulesW(NWChar * * pSzValue);
```

Parameters

Parameters	Description
NWChar * * pSzValue	[out] A string that contains the list of scanners separated by semicolons.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

This function is a Unicode version of the function.

4.1.1.7 NffvGetErrorMessageA Function

Gets an error message. Use this function for errors handling.

C++

```
NInt N_API NffvGetErrorMessageA(NResult code, NAChar * szValue);
```

Parameters

Parameters	Description
NResult code	[in] An error code.
NAChar * szValue	[out] Pointer to memory block that contains an error description.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

This function is an ANSI version of the function.

4.1.1.8 NffvGetErrorMessageW Function

Gets an error message. Use this function for errors handling.

C++

```
NInt N_API NffvGetErrorMessageW(NResult code, NWChar * szValue);
```

Parameters

Parameters	Description
NResult code	[in] An error code.
NWChar * szValue	[out] Pointer to memory block that contains an error description.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

This function is a Unicode version of the function.

4.1.1.9 NffvGetMatchingThreshold Function

Gets the minimum similarity value that verification function uses to determine whether the fingerprint matches.

C++

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

Parameters

Parameters	Description
NInt * pValue	[out] Similarity value (matching threshold) for the Nffv. Values are in range [0, MaxInt]. MaxInt is a maximum integer value.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

For more information about the matching threshold, please read chapter Matching Threshold (see page 15).

4.1.1.10 NffvGetQualityThreshold Function

Gets an image quality threshold.

C++

```
NResult N_API NffvGetQualityThreshold(NByte * pValue);
```

Parameters

Parameters	Description
NByte * pValue	[out] Quality threshold. The value is in range [0, 255].

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

For more information about the quality threshold, please read chapter Quality Threshold (see page 15).

4.1.1.11 NffvGetUser Function

Gets the information from a users list about an enrolled user.

C++

```
NResult N_API NffvGetUser(NInt index, HNffvUser * pValue);
```

Parameters

Parameters	Description
NInt index	[in] An index of a user who was enrolled to a database.
HNffvUser * pValue	[out] Information about an enrolled user.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

To get an index of a user you can use NffvGetUserIndexById (see page 27) function.

4.1.1.12 NffvGetUserById Function

Returns user details by the Id from a database.

C++

```
NResult N_API NffvGetUserById(NInt id, HNffvUser * pValue);
```

Parameters

Parameters	Description
NInt id	[in] User's identification number in a database. This Id is always unique.
HNffvUser * pValue	[out] Information about a user who was enrolled to a database.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

4.1.1.13 NffvGetUserCount Function

Retrieves the number of users enrolled to database.

C++

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

Parameters

Parameters	Description
NInt * pValue	[out] The number of enrolled users.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

4.1.1.14 NffvGetUserIndexById Function

Retrieves the index from users list of a user indicated by the Id.

C++

```
NResult N_API NffvGetUserIndexById(NInt id, NInt * pValue);
```

Parameters

Parameters	Description
NInt id	[in] The user Id. This Id is used in a users database.
NInt * pValue	[out] An index of a user.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

4.1.1.15 NffvInitializeA Function

Initializes the FFV. This function as a parameters takes a name and a password of a previously created database or creates a new database.

C++

```
NResult N_API NffvInitializeA(const NAChar * szDbName, const NAChar * szPassword, const
NAChar * szScannerModules);
```

Parameters

Parameters	Description
const NAChar * szDbName	[out] The name of a database. This database will be used to save user fingerprints. The database will be saved to a working folder (or other folder) as a file.
const NAChar * szPassword	[out] A database password. If you don't want to protect a database by password, use a blank a password.
const NAChar * szScannerModules	[out] A list of scanner modules that should be loaded. It is a list of fingerprint scanners that you will use in your application. If the value is an empty string then no scanners are loaded. If the value is null all scanner modules are loaded. Each scanner module in a list should be separated by a semicolon.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

This function is an ANSI version of the function.

4.1.1.16 NffvInitializeW Function

Initializes the FFV. This function as a parameters takes a name and a password of a previously created database or creates a new database.

C++

```
NResult N_API NffvInitializeW(const NWChar * szDbName, const NWChar * szPassword, const
NWChar * szScannerModules);
```

Parameters

Parameters	Description
const NWChar * szDbName	[out] The name of a database. This database will be used to save user fingerprints. The database will be saved to a working folder (or other folder) as a file.
const NWChar * szPassword	[out] A database password. If you don't want to protect a database by password, use a blank a password.

const NWChar * szScannerModules	[out] A list of scanner modules that should be loaded. It is a list of fingerprint scanners that you will use in your application. If the value is an empty string then no scanners are loaded. If the value is null all scanner modules are loaded. Each scanner module in a list should be separated by a semicolon.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

This function is Unicode version of the function.

4.1.1.17 NffvRemoveUser Function

Removes a user from users list (database).

C++

```
NResult N_API NffvRemoveUser(NInt index);
```

Parameters

Parameters	Description
NInt index	[in] An index number of a user that should be removed from a list.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

All enrolled users during the execution of an application are loaded from a database to a list.

4.1.1.18 NffvSetMatchingThreshold Function

Sets the minimum similarity value that verification function uses to determine whether the fingerprint matches.

C++

```
NResult N_API NffvSetMatchingThreshold(NInt value);
```

Parameters

Parameters	Description
NInt value	[in] Similarity value (matching threshold) to set.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

For more information about the matching threshold, please read chapter Matching Threshold (see page 15).

The default matching threshold value is 48.

4.1.1.19 NffvSetQualityThreshold Function

Sets an image quality threshold.

C++

```
NResult N_API NffvSetQualityThreshold(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] Quality threshold to set.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Remarks

For more information about the quality threshold, please read chapter Quality Threshold (see page 15).

The default matching threshold value is 100.

4.1.1.20 NffvUninitialize Function

Releases memory resources.

C++

```
void N_API NffvUninitialize();
```

Returns

If the functions succeeds the return value is N_OK. Otherwise, the function returns an error code (see page 80).

4.1.1.21 NffvUserGetHBitmap Function

Gets a handle to the bitmap of a user fingerprint.

C++

```
NResult N_API NffvUserGetHBitmap(HNffvUser hUser, NHandle * pHBitmap);
```

Parameters

Parameters	Description
HNffvUser hUser	[in] A handle to NffvUser (see page 32) object which is used to manage users.
NHandle * pHBitmap	[out] A handle to a bitmap of the last scanned fingerprint.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

4.1.1.22 NffvUserGetImage Function

Gets a user's fingerprint image which was enrolled to a database.

C++

```
NResult N_API NffvUserGetImage(HNffvUser hUser, NUInt * pWidth, NUInt * pHeight, NFloat *
pHorzResolution, NFloat * pVertResolution, NSizeType * pStride, void * pPixels);
```

Parameters

Parameters	Description
HNffvUser hUser	[in] A handle to user (a user who was enrolled to a database).
NUInt * pWidth	[out] The width of an image.
NUInt * pHeight	[out] The height of an image.

NFloat * pHorzResolution	[out] The horizontal resolution of an image.
NFloat * pVertResolution	[out] The vertical resolution of an image.
NSizeType * pStride	[out] The stride of a fingerprint image. Stride of the image depends on image pixel format and width.
void * pPixels	[out] An image pixel format. If the value is Null then a width, height, resolution and stride of an image is returned. When you have these values you can allocate memory buffer for user image. The size of memory buffer can be calculated using this formula: height * stride.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

4.1.1.23 NffvVerify Function

Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

C++

```
NResult N_API NffvVerify(HNffvUser hUser, NUInt timeout, NffvStatus * pStatus, NInt *
pScore);
```

Parameters

Parameters	Description
HNffvUser hUser	[in] A handle to a database record that should be matched with the scanned fingerprint.
NUInt timeout	[in] Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than *timeout* milliseconds.
NffvStatus * pStatus	[out] One of the verification status values enumerated in NffvStatus (see page 32).
NInt * pScore	[out] A matching score of two fingerprints verification.

Returns

If the function succeeds the return value is N_OK. Otherwise, an error code (see page 80) is returned.

Example

This C++ example demonstrates how to verify two fingerprints:

```
UINT CMainForm::VerifyUserThread(LPVOID pParam)
{
    VerifyParam *pVerifyParam = (VerifyParam*)pParam;
    CMainForm* form = ((CMainForm*)pVerifyParam->pMainForm);

    NResult result = NffvVerify(pVerifyParam->hUser, 20000,
                     &pVerifyParam->engineStatus, pVerifyParam->pScore);
    if(NFailed(result)) throw result;

    pVerifyParam->pBusyForm->Stop();

    return 0;   // thread completed successfully
}
```

4.1.2 Types

The following table lists types in this documentation.

4

Module

C/C++ Reference (see page 21)

Types

Name	Description
NffvStatus (see page 32)	Enumerates enrollment or verification values of the Nffv (see page 61).

4.1.2.1 NffvStatus Enumeration

Enumerates enrollment or verification values of the Nffv (see page 61).

C++

```
typedef enum NffvStatus_ NffvStatus;
```

4.1.3 Macros

The following table lists macros in this documentation.

Macros

Name	Description
NFFV_MAX_USER_COUNT (see page 32)	The maximum number of users that can be enrolled to a database.

Module

C/C++ Reference (see page 21)

4.1.3.1 NFFV_MAX_USER_COUNT Macro

The maximum number of users that can be enrolled to a database.

C++

```
#define NFFV_MAX_USER_COUNT 10
```

Notes

You can not change this value.

4.2 .NET Reference

This chapter provides the Free Fingerprint Verification SDK programming reference for Microsoft .NET framework.

Remarks

If you are developing your own application using one of a .NET programming language, you should include this dynamic-link library into your biometric solution project:

- Neurotec.Biometrics.Nffv.dll (contains enrollment and verification methods). This Dll is a wrapper of the Nffv.dll.

Nffv.dll file should be located in the same folder as Neurotec.Biometrics.Nffv.dll. Also NffvServer.exe is required for using Neurotec.Biometrics.Nffv.dll.

Namespaces

Name	Description
Neurotec.Biometrics (see page 33)	Contains classes and methods that provide the Free Fingerprint Verification SDK functionality.

4.2.1 Neurotec.Biometrics Namespace

Contains classes and methods that provide the Free Fingerprint Verification SDK functionality.

Module

.NET Reference (see page 32)

Classes

	Name	Description
	Nffv (see page 33)	The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with user collection and enrolling or verifying user fingerprints.
	NffvUser (see page 40)	Provides methods and properties for working with users.

Structs, Records, Enums

	Name	Description
	NffvStatus (see page 41)	Enumerates enrollment or verification status values.

4.2.1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	Nffv (see page 33)	The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with user collection and enrolling or verifying user fingerprints.
	NffvUser (see page 40)	Provides methods and properties for working with users.

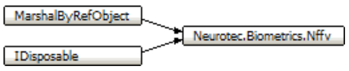
4.2.1.1.1 Nffv Class

The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with user collection and enrolling or verifying user fingerprints.

C#

```
public class Nffv : MarshalByRefObject, IDisposable;
```

Class Hierarchy



Methods

	Name	Description
	Nffv (see page 34)	Initializes a new instance of the Nffv class. During the initialization a new database is created or used previously created.

Nffv Classes

	Name	Description
	UserCollection (see page 35)	Represents a collection of NffvUsers objects that represent the user fingerprints enrolled to a database.

Nffv Fields

	Name	Description
	DllName (see page 37)	The name of a dynamic-linked library which contains unmanaged functionality of the Free Fingerprint Verification SDK.
	MaxUserCount (see page 37)	The maximum number of users that can be enrolled to a database.

Nffv Methods

	Name	Description
	Cancel (see page 37)	Cancels a fingerprint enrollment or verification operation.
	Dispose (see page 38)	Disposes resources used by the Nffv.
	Enroll (see page 38)	Gets a fingerprint from a scanner and saves it to a database.
S	GetAvailableScannerModules (see page 39)	Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.
	GetUserById (see page 39)	Returns a user details by the Id from the UserCollection (see page 35).
	Verify (see page 39)	Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

Nffv Properties

	Name	Description
	MatchingThreshold (see page 40)	Gets or sets the minimum similarity value that verification method uses to determine whether the fingerprint matches.
	QualityThreshold (see page 40)	Gets or sets image quality threshold.

4.2.1.1.1.1 Nffv Constructor

4.2.1.1.1.1.1 Nffv.Nffv Constructor (string, string)

Initializes a new instance of the Nffv class. During the initialization a new database is created or used previously created.

C#

```
public Nffv(string dbName, string password);
```

Parameters

Parameters	Description
string dbName	A name of database. This database will be used to save user fingerprints. The database will be saved to a working folder as a file.
string password	A database password. If you don't want to protect a database by password, use an empty string as a password.

Example

This C# example code demonstrates how to create a new instance of the Nffv calss.

```
string dbName = "FingerprintsDatabase.dat";
string password = "passwd";

Neurotec.Biometrics.Nffv engine = null;

//Creates a new instance of the Nffv class
engine = new Neurotec.Biometrics.Nffv(dbName, password);
```

The same example code for VB.NET:

```
Dim dbName As String = "FingerprintsDatabase.dat"
Dim password As String = "passwd"

Dim engine As Global.Neurotec.Biometrics.Nffv = Nothing

engine = New Global.Neurotec.Biometrics.Nffv(dbName, password)
```

4.2.1.1.1.1.2 Nffv.Nffv Constructor (string, string, string)

Initializes a new instance of the Nffv class. During the initialization a new database is created or used previously created.

C#

```
public Nffv(string dbName, string password, string scannerModules);
```

Parameters

Parameters	Description
string dbName	A name of database. This database will be used to save user fingerprints. The database will be saved to a working folder as a file.
string password	A database password. If you don't want to protect a database by password, use an empty string as a password.
string scannerModules	A list of scanner modules that should be loaded. It is a list of fingerprint scanners that you will use in your application. Each fingerprint scanner's name in the scanner module is separated by semicolon.

Remarks

For the list of available fingerprint scanners see a chapter Supported Scanners.

Example

This C# example code demonstrates how to create a new instance of the Nffv calss.

```
string dbName = "FingerprintsDatabase.dat";
string password = "passwd";
string scanners = "Upek;Futronic";

Neurotec.Biometrics.Nffv engine = null;

//Creates a new instance of the Nffv class
engine = new Neurotec.Biometrics.Nffv(dbName, password, scanners);
```

The same example code for VB.NET:

```
Dim dbName As String = "FingerprintsDatabase.dat"
Dim password As String = "passwd"
Dim scanners As String = "Upek;Futronic"

Dim engine As Global.Neurotec.Biometrics.Nffv = Nothing

engine = New Global.Neurotec.Biometrics.Nffv(dbName, password, scanners)
```

4.2.1.1.1.2 Nffv Classes

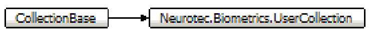
4.2.1.1.1.2.1 Nffv.UserCollection Class

Represents a collection of NffvUsers objects that represent the user fingerprints enrolled to a database.

C#

```
[Serializable]
public sealed class UserCollection : CollectionBase;
```

Class Hierarchy



Notes

This class is a sealed class, so it has a limited extensibility (other classes cannot inherit from it).

UserCollection Methods

	Name	Description
	Add (see page 36)	Adds a user to a UserCollection.
	Contains (see page 36)	Returns a Boolean value indicating whether a UserCollection object contains an element with a specified key.
	IndexOf (see page 37)	Returns an index of the UserCollection item specified by Id.

4.2.1.1.1.2.1.1 UserCollection Methods

4.2.1.1.1.2.1.1.1 Nffv.UserCollection.Add Method

Adds a user to a UserCollection (see page 35).

C#

```
internal NffvUser Add(IntPtr hUser);
```

Parameters

Parameters	Description
IntPtr hUser	A reference to an object that represents a user which should be added to a collection.

Example

To add a user to database you can use this C# code:

```
public class UserEnrollment
{
    UserDatabase _userDB;
    _userDB.Add(new UserRecord(engineUser.Id, userName));
}

public class UserRecord
{
   //...
   public UserRecord(int id, string name)
   {
      _id = id;
      _name = name;
   }
}
```

4.2.1.1.1.2.1.1.2 Nffv.UserCollection.Contains Method

Returns a Boolean value indicating whether a UserCollection (see page 35) object contains an element with a specified key.

C#

```
public bool Contains(int id);
```

Parameters

Parameters	Description
int id	An integer value that specifies the Id for which to search the element of the collection.

Returns

A Boolean value indicating whether the UserCollection (see page 35) contains an elements with the specified Id.

If the return value is True, the collection contains an element with an Id specified. Otherwise, the return value is False.

Example

This C# example demonstrates how to use this method:

```
int id = 3;

if UserCollection.Contains(id)
   MsgBox("The desired user is in collection");
else
   MsgBox("The desired user was not find in a collection");
```

The VB.NET code this method:

```
Dim id As Integer = 3

If UserCollection.Contains(id) Then
    MsgBox("The desired user is in the collection.")
Else
    MsgBox("The desired user was not find in the collection.")
End If
```

4.2.1.1.1.2.1.1.3 Nffv.UserCollection.IndexOf Method

Returns an index of the UserCollection (see page 35) item specified by Id.

C#

```
public int IndexOf(int id);
```

Parameters

Parameters	Description
int id	The Id of a user to search in a collection.

Returns

A collection index of a user specified by Id.

4.2.1.1.1.3 Nffv Fields

4.2.1.1.1.3.1 Nffv.DllName Field

The name of a dynamic-linked library which contains unmanaged functionality of the Free Fingerprint Verification SDK.

C#

```
public const string DllName = "Nffv.dll";
```

4.2.1.1.1.3.2 Nffv.MaxUserCount Field

The maximum number of users that can be enrolled to a database.

C#

```
public const int MaxUserCount = 10;
```

Remarks

You can add up-to 10 users to a database.

4.2.1.1.1.4 Nffv Methods

4.2.1.1.1.4.1 Nffv.Cancel Method

Cancels a fingerprint enrollment or verification operation.

C#

```
public void Cancel();
```

Remarks

This method is useful when the fingerprint enrollment or verification operation take too long. In this case a message box can be shown for a user to cancel this operation.

Example

This C# code demonstrates how to cancel enrollment or verification operation:

```
Nffv engine;
```

```
engine.Cancel();
```

The same code using VB.NET notation:

```
Private engine As Nffv
```

```
engine.Cancel()
```

4.2.1.1.1.4.2 Nffv.Dispose Method

Disposes resources used by the Nffv (see page 33).

C#

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

4.2.1.1.1.4.3 Nffv.Enroll Method

Gets a fingerprint from a scanner and saves it to a database.

C#

```
public NffvUser Enroll(uint timeout, out NffvStatus status);
```

Parameters

Parameters	Description
uint timeout	Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than *timeout* milliseconds.
out NffvStatus status	Enrollment (see page 14) status value indicated by one of the value enumerated in NffvStatus (see page 41).

Returns

A reference to NffvUser (see page 40) object which provides methods for managing enrolled users.

If there were problem enrolling a fingerprint, the method returns a zero pointer.

Example

This C# example demonstrates the usage of the *Enroll* method:

```
//Field that holds a reference to Nffv object
Nffv engine;

//Internal class that saves the result of fingerprint enrolment
internal class EnrollmentResult
{
    public NffvStatus engineStatus;
    public NffvUser engineUser;
};

//Method used for a fingerprint enrollment
public void doEnroll(object sender, DoWorkEventArgs args)
{
    EnrollmentResult enrollmentResults = new EnrollmentResult();
    enrollmentResults.engineUser = engine.Enroll(20000, out enrollmentResults.engineStatus);
    args.Result = enrollmentResults;
}
```

4.2.1.1.1.4.4 Nffv.GetAvailableScannerModules Method

Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.

C#

```
public static string GetAvailableScannerModules();
```

Returns

A string that contains the list of scanners separated by semicolons.

4.2.1.1.1.4.5 Nffv.GetUserById Method

Returns a user details by the Id from the UserCollection (see page 35).

C#

```
public NffvUser GetUserById(int id);
```

Parameters

Parameters	Description
int id	User's identification number in a collection.

Returns

A reference to the NffvUser (see page 40) object that contains an information about a user indicated by Id.

4.2.1.1.1.4.6 Nffv.Verify Method

Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

C#

```
public int Verify(NffvUser user, uint timeout, out NffvStatus status);
```

Parameters

Parameters	Description
NffvUser user	A reference to a database record that should be matched with the scanned fingerprint.
uint timeout	Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than *timeout* milliseconds.
out NffvStatus status	The verification status value indicated by one of the value enumerated in NffvStatus (see page 41).

Returns

This function returns a matching score.

Example

This C# sample code demonstrates how to verify two fingerprints.

```
Nffv engine;

//An internal class that saves the verification result
internal class VerificationResult
{
    public NffvStatus engineStatus;
    public int score;
};

public void doVerify(object sender, DoWorkEventArgs args)
{
    VerificationResult verificationResult = new VerificationResult();
    verificationResult.score = engine.Verify((NffvUser)args.Argument, 20000, out
verificationResult.engineStatus);
    args.Result = verificationResult;
}
```

Note that it isn't a complete code that can be used in your application.

For a complete code see the C# Sample application.

4.2.1.1.1.5 Nffv Properties

4.2.1.1.1.5.1 Nffv.MatchingThreshold Property

Gets or sets the minimum similarity value that verification method uses to determine whether the fingerprint matches.

C#

```
public int MatchingThreshold;
```

Property value

The minimum similarity value that verification function accept for the same finger fingerprints. The default value is 0.01 %.

4.2.1.1.1.5.2 Nffv.QualityThreshold Property

Gets or sets image quality threshold.

C#

```
public byte QualityThreshold;
```

Property value

The fingerprint quality threshold. The value should be in range [0, 255]. The default value is 100.

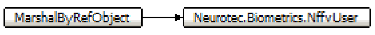
4.2.1.1.2 NffvUser Class

Provides methods and properties for working with users.

C#

```
public sealed class NffvUser : MarshalByRefObject;
```

Class Hierarchy



NffvUser Methods

	Name	Description
◆	GetBitmap (↗ see page 40)	Returns the bitmap of the last scanned fingerprint.
◆	GetHBitmap (↗ see page 41)	Returns a handle to the bitmap of the last scanned fingerprint.

4.2.1.1.2.1 NffvUser Methods

4.2.1.1.2.1.1 NffvUser.GetBitmap Method

Returns the bitmap of the last scanned fingerprint.

C#

```
public Bitmap GetBitmap();
```

Returns

A Bitmap object.

4.2.1.1.2.1.2 NffvUser.GetHBitmap Method

Returns a handle to the bitmap of the last scanned fingerprint.

C#

```
public IntPtr GetHBitmap();
```

Returns

A pointer to Bitmap object.

4.2.1.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	NffvStatus (see page 41)	Enumerates enrollment or verification status values.

4.2.1.2.1 Neurotec.Biometrics.NffvStatus Enumeration

Enumerates enrollment or verification status values.

C#

```
[Serializable]
public enum NffvStatus {
  TemplateCreated = 1,
  NoScanner = 2,
  ScannerTimeout = 3,
  UserCanceled = 4,
  QualityCheckFailed = 100
}
```

Members

Members	Description
TemplateCreated = 1	Indicates that the fingerprint template was created.
NoScanner = 2	Indicates that there is no fingerprint scanner connected.
ScannerTimeout = 3	Indicates that the fingerprint scanner has reached the timeout.
UserCanceled = 4	Indicates that a user has canceled a fingerprint scanning.
QualityCheckFailed = 100	Indicates that the Free Fingerprint Verification SDK had failed to check the quality of a fingerprint.

4.3 Java Reference

This chapter provides the Free Fingerprint Verification SDK programming reference for Java programming language.

Notes

You can find source files of the Java wrapper under the FFV SDK directory *\samples\Java\NffvJavaWrapper\src\com\neurotechnology*.

Packages

Name	Description
com.neurotechnology.Library (see page 42)	Classes under this namespace provides methods for working with com.neurotechnology.Nffv (see page 51) library.
com.neurotechnology.Nffv (see page 51)	Classes under this namespace provides methods for the com.neurotechnology.Nffv (see page 52) library.

4.3.1 com.neurotechnology.Library Package

Classes under this namespace provides methods for working with com.neurotechnology.Nffv (see page 51) library.

Module

Java Reference (see page 41)

Classes

	Name	Description
✦	LibraryInfo (see page 43)	Provides methods for getting a library information.
✦	NativeManager (see page 44)	This class is responsible for loading Neurotechnology modules and native library that contains implementation of native methods in JavaWrapper classes.
✦	NativeObject (see page 46)	Provides methods for working with native objects.
✦	NetInstall (see page 47)	NetInstall class manages installation of Neurotechnogy modules for Applet applications. Since com.neurotechnology.Nffv (see page 51) classes are using native libraries these libraries need to be accessible for application that are using classes from com.neurotechnology.Nffv (see page 51). This class parses files that consist list of libraries needed and allows download them to predefined location.
✦	ScannerFiles (see page 49)	Provides methods for managing scanner files.
✦	TemplateFileFilter (see page 50)	Extends Java's FileFilter interface. Provides methods for filtering files.

4.3.1.1 Classes

The following table lists classes in this documentation.

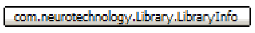
Classes

	Name	Description
✦	LibraryInfo (see page 43)	Provides methods for getting a library information.
✦	NativeManager (see page 44)	This class is responsible for loading Neurotechnology modules and native library that contains implementation of native methods in JavaWrapper classes.
✦	NativeObject (see page 46)	Provides methods for working with native objects.
✦	NetInstall (see page 47)	NetInstall class manages installation of Neurotechnogy modules for Applet applications. Since com.neurotechnology.Nffv (see page 51) classes are using native libraries these libraries need to be accessible for application that are using classes from com.neurotechnology.Nffv (see page 51). This class parses files that consist list of libraries needed and allows download them to predefined location.
✦	ScannerFiles (see page 49)	Provides methods for managing scanner files.
✦	TemplateFileFilter (see page 50)	Extends Java's FileFilter interface. Provides methods for filtering files.

4.3.1.1.1 LibraryInfo Class

Provides methods for getting a library information.

Class Hierarchy



Java

```
public class LibraryInfo;
```

LibraryInfo Methods

	Name	Description
◆	getCompany (↗ see page 43)	Gets a company name.
◆	getCopyright (↗ see page 43)	Gets a copyright notice from the library.
◆	getProduct (↗ see page 43)	Gets product name.
◆	getTitle (↗ see page 43)	Gets a title form the library.
◆	getVersionBuild (↗ see page 44)	Gets library build version.
◆	getVersionMajor (↗ see page 44)	Gets library's major version.
◆	getVersionMinor (↗ see page 44)	Gets library's minor version.
◆	getVersionRevision (↗ see page 44)	Gets library's revision version.

4.3.1.1.1.1 LibraryInfo Methods

4.3.1.1.1.1.1 LibraryInfo.getCompany Method

Gets a company name.

Returns

A string that contains a company name.

Java

```
public String getCompany();
```

4.3.1.1.1.1.2 LibraryInfo.getCopyright Method

Gets a copyright notice from the library.

Returns

A string that contains library's copyright notice.

Java

```
public String getCopyright();
```

4.3.1.1.1.1.3 LibraryInfo.getProduct Method

Gets product name.

Returns

A string that contains product name.

Java

```
public String getProduct();
```

4.3.1.1.1.1.4 LibraryInfo.getTitle Method

Gets a title form the library.

Returns

A string that contains library title.

Java

```
public String getTitle();
```

4.3.1.1.1.1.5 LibraryInfo.getVersionBuild Method

Gets library build version.

Returns

Library's build version number.

Java

```
public int getVersionBuild();
```

4.3.1.1.1.1.6 LibraryInfo.getVersionMajor Method

Gets library's major version.

Returns

Major version of a library..

Java

```
public int getVersionMajor();
```

4.3.1.1.1.1.7 LibraryInfo.getVersionMinor Method

Gets library's minor version.

Returns

Library's minor version number.

Java

```
public int getVersionMinor();
```

4.3.1.1.1.1.8 LibraryInfo.getVersionRevision Method

Gets library's revision version.

Returns

Library's revision version number.

Java

```
public int getVersionRevision();
```

4.3.1.1.2 NativeManager Class

This class is responsible for loading Neurotechnology modules and native library that contains implementation of native methods in JavaWrapper classes.

Class Hierarchy

Java

```
public class NativeManager;
```

NativeManager Fields

	Name	Description
◆	defaultlibrary (see page 45)	Default name of a library.

	isLibraryLoaded (see page 45)	A boolean value indicating if a library was loaded.

NativeManager Methods

	Name	Description
	getProductName (see page 45)	Gets a product name. If a library fails to load an exception is thrown.
	getVersionMajor (see page 45)	Gets a major version of a library. If a library fails to load an exception is thrown.
	getVersionMinor (see page 45)	Gets a minor version of a library. If a library fails to load an exception is thrown.
	getWrapperLibraryInfo (see page 46)	Gets information (such as company name, product, copyright notice) about wrapper's library.
	isLoaded (see page 46)	Checks if a library was loaded to memory.
	loadDefault (see page 46)	Loads a default library.
	loadFile (see page 46)	Loads default and Java native libraries.

4.3.1.1.2.1 NativeManager Fields

4.3.1.1.2.1.1 NativeManager.defaultlibrary Field

Default name of a library.

Java

```
public static String defaultlibrary = "NeurotecJavaNative";
```

4.3.1.1.2.1.2 NativeManager.isLibraryLoaded Field

A boolean value indicating if a library was loaded.

Java

```
protected static boolean isLibraryLoaded;
```

4.3.1.1.2.2 NativeManager Methods

4.3.1.1.2.2.1 NativeManager.getProductName Method

Gets a product name. If a library fails to load an exception is thrown.

Returns

String that contains a product name.

Java

```
public static String getProductName() throws Exception;
```

4.3.1.1.2.2.2 NativeManager.getVersionMajor Method

Gets a major version of a library. If a library fails to load an exception is thrown.

Returns

Integer value of library version.

Java

```
public static int getVersionMajor() throws Exception;
```

4.3.1.1.2.2.3 NativeManager.getVersionMinor Method

Gets a minor version of a library. If a library fails to load an exception is thrown.

Returns

Integer value of library minor version.

Java

```
public static int getVersionMinor() throws Exception;
```

4.3.1.1.2.2.4 NativeManager.getWrapperLibraryInfo Method

Gets information (such as company name, product, copyright notice) about wrapper's library.

Returns

LibraryInfo (see page 43) object that information about wrapper.

Java

```
public static LibraryInfo getWrapperLibraryInfo();
```

4.3.1.1.2.2.5 NativeManager.isLoaded Method

Checks if a library was loaded to memory.

Returns

Boolean value indicating if a library was loaded to memory.

Java

```
public static boolean isLoaded();
```

4.3.1.1.2.2.6 NativeManager.loadDefault Method

Loads a default library.

Java

```
public static void loadDefault();
```

4.3.1.1.2.2.7 NativeManager.loadFile Method

Loads default and Java native libraries.

Parameters

Parameters	Description
String nlicensing	Name of the Neurotechnology's NLicensing library.
neurotecjavanative	Name of a Java native library.

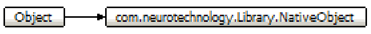
Java

```
public static void loadFile(String neuratecjavanative, String nlicensing);
```

4.3.1.1.3 NativeObject Class

Provides methods for working with native objects.

Class Hierarchy



Java

```
public class NativeObject extends Object;
```

Methods

	Name	Description
◆	NativeObject (see page 47)	Creates a new instance of the NativeObject.

NativeObject Methods

	Name	Description
◆	getHandle (see page 47)	Gets a handle of the NativeObject.
◆	setHandle (see page 47)	Sets a handle for the NativeObject.

4.3.1.1.3.1 NativeObject.NativeObject Constructor

Creates a new instance of the NativeObject.

Java

```
public NativeObject();
```

4.3.1.1.3.2 NativeObject Methods

4.3.1.1.3.2.1 NativeObject.getHandle Method

Gets a handle of the NativeObject (see page 46).

Returns

Handle to the NativeObject (see page 46).

Java

```
public long getHandle();
```

4.3.1.1.3.2.2 NativeObject.setHandle Method

Sets a handle for the NativeObject (see page 46).

Parameters

Parameters	Description
long handle	Handle for the NativeObject (see page 46).

Java

```
public void setHandle(long handle);
```

4.3.1.1.4 NetInstall Class

NetInstall class manages installation of Neurotechnogy modules for Applet applications. Since com.neurotechnology.Nffv (see page 51) classes are using native libraries these libraries need to be accessible for application that are using classes from com.neurotechnology.Nffv (see page 51). This class parses files that consist list of libraries needed and allows download them to predefined location.

Class Hierarchy

Java

```
public class NetInstall;
```

Methods

	Name	Description
◆	NetInstall (see page 48)	Creates a new instance of the NetInstall.

NetInstall Methods

	Name	Description
◆	checkLoadDefault (see page 48)	Tries to load libraries by default load method. Search is done in system path and user path variables. If Libraries are found they are loaded and checkLoadDefault returns true. Overwise it returns false.
◆	checkLoadTemp (see page 48)	Tries to load libraries from temporary folder that is located in /.neurotec/.If load is successful returns true and false overwise.
◆	getEnvironment (see page 48)	Gets an environment properties.
◆	getMainLibrariesLinux (see page 48)	Retrieves an array of objects (vector) of libraries for fingerprint scanners. This method returns libraries for Linux.

◆	getMainLibrariesWindows (see page 48)	Retrieves an array of objects (vector) of libraries for Windows OS. These libraries are used by the FFV SDK.
◆	getScannerLibrariesWindows (see page 49)	Retrieves an array of objects (vector) of libraries for fingerprint scanners.
◆	installTemp (see page 49)	Installs Neurotec libraries to temporary directory /.neurotec/

4.3.1.1.4.1 NetInstall.NetInstall Constructor

Creates a new instance of the NetInstall.

Java

```
public NetInstall() throws Exception;
```

4.3.1.1.4.2 NetInstall Methods

4.3.1.1.4.2.1 NetInstall.checkLoadDefault Method

Tries to load libraries by default load method. Search is done in system path and user path variables. If Libraries are found they are loaded and checkLoadDefault returns true. Overwise it returns false.

Returns

Boolean value indicating if the default FFV SDK library was loaded.

Java

```
public static boolean checkLoadDefault();
```

4.3.1.1.4.2.2 NetInstall.checkLoadTemp Method

Tries to load libraries from temporary folder that is located in /.neurotec/.If load is successful returns true and false overwise.

Java

```
public boolean checkLoadTemp();
```

4.3.1.1.4.2.3 NetInstall.getEnvironment Method

Gets an environment properties.

Returns

Property list that contains environment data.

Java

```
public Properties getEnvironment() throws java.io.IOException;
```

4.3.1.1.4.2.4 NetInstall.getMainLibrariesLinux Method

Retrieves an array of objects (vector) of libraries for fingerprint scanners. This method returns libraries for Linux.

Returns

Array of objects (vector) that contains strings of libraries for the Linux.

Java

```
public Vector<String> getMainLibrariesLinux() throws Exception;
```

4.3.1.1.4.2.5 NetInstall.getMainLibrariesWindows Method

Retrieves an array of objects (vector) of libraries for Windows OS. These libraries are used by the FFV SDK.

Returns

Array of objects (vector) that contains strings of libraries for the Windows.

Java

```
public Vector<String> getMainLibrariesWindows() throws Exception;
```

4.3.1.1.4.2.6 NetInstall.getScannerLibrariesWindows Method

Retrieves an array of objects (vector) of libraries for fingerprint scanners.

Returns

Array of objects (vector) that contains fingerprint scanners names.

Java

```
public Vector<ScannerFiles> getScannerLibrariesWindows() throws Exception;
```

4.3.1.1.4.2.7 NetInstall.installTemp Method

Installs Neurotec libraries to temporary directory /.neurotec/

Java

```
public void installTemp(String codeBase, Vector<String> mainlibs, Vector<ScannerFiles>
scanners);
```

4.3.1.1.5 ScannerFiles Class

Provides methods for managing scanner files.

Class Hierarchy

com.neurotechnology.Library.ScannerFiles

Java

```
public class ScannerFiles;
```

Methods

	Name	Description
◆	ScannerFiles (see page 49)	Creates a new instance of the ScannerFiles.

ScannerFiles Methods

	Name	Description
◆	addFile (see page 49)	Adds a file which is used by a fingerprint scanner.
◆	getFiles (see page 50)	Returns an array of objects (vector) that contains names of all fingerprint scanners files.
◆	getName (see page 50)	Gets a name of a fingerprint scanner.
◆	setName (see page 50)	Sets a scanner name.

4.3.1.1.5.1 ScannerFiles.ScannerFiles Constructor

Creates a new instance of the ScannerFiles.

Java

```
protected ScannerFiles();
```

4.3.1.1.5.2 ScannerFiles Methods

4.3.1.1.5.2.1 ScannerFiles.addFile Method

Adds a file which is used by a fingerprint scanner.

Parameters

Parameters	Description
String fileName	Name of a file to add.

Java

```java
protected void addFile(String fileName);
```

4.3.1.1.5.2.2 ScannerFiles.getFiles Method

Returns an array of objects (vector) that contains names of all fingerprint scanners files.

Returns

Vector that contains files names.

Java

```java
public Vector<String> getFiles();
```

4.3.1.1.5.2.3 ScannerFiles.getName Method

Gets a name of a fingerprint scanner.

Returns

String that contains a name of fingerprint scanner.

Java

```java
public String getName();
```

4.3.1.1.5.2.4 ScannerFiles.setName Method

Sets a scanner name.

Parameters

Parameters	Description
String name	Name of a fingerprint scanner to set.

Java

```java
protected void setName(String name);
```

4.3.1.1.6 TemplateFileFilter Class

Extends Java's FileFilter interface. Provides methods for filtering files.

Class Hierarchy

Java

```java
public class TemplateFileFilter extends FileFilter;
```

TemplateFileFilter Methods

	Name	Description
	accept (see page 50)	Tests whether or not the specified file should be included in a file list.
	getDescription (see page 51)	Gets a description of template files.
	getFileExtension (see page 51)	Gets the extension of a file.

4.3.1.1.6.1 TemplateFileFilter Methods

4.3.1.1.6.1.1 TemplateFileFilter.accept Method

Tests whether or not the specified file should be included in a file list.

Parameters

Parameters	Description
File f	Path to a file that should be tested.

Returns

Boolean value that indicates if file should be be included. File is included when a return value is true.

Java

```
public boolean accept(File f);
```

4.3.1.1.6.1.2 TemplateFileFilter.getDescription Method

Gets a description of template files.

Returns

String that contains template files description.

Java

```
public String getDescription();
```

4.3.1.1.6.1.3 TemplateFileFilter.getFileExtension Method

Gets the extension of a file.

Parameters

Parameters	Description
File f	Path to a file which extension should be returned.

Returns

String that contains file extension.

Java

```
public static String getFileExtension(File f);
```

4.3.2 com.neurotechnology.Nffv Package

Classes under this namespace provides methods for the com.neurotechnology.Nffv (see page 52) library.

Module

Java Reference (see page 41)

Classes

	Name	Description
	Nffv (see page 52)	The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with users list and enrolling or verifying user fingerprints.
	NffvImage (see page 56)	Provides methods for managing images.
	NffvUser (see page 59)	Provides methods for working with users.
	ScannerModule (see page 60)	Provides methods for setting and getting scanner names from the ScannerModule.

4.3.2.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	Nffv (⧉ see page 52)	The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with users list and enrolling or verifying user fingerprints.
	NffvImage (⧉ see page 56)	Provides methods for managing images.
	NffvUser (⧉ see page 59)	Provides methods for working with users.
	ScannerModule (⧉ see page 60)	Provides methods for setting and getting scanner names from the ScannerModule.

4.3.2.1.1 Nffv Class

The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with users list and enrolling or verifying user fingerprints.

Class Hierarchy

Java

```
public class Nffv;
```

Methods

	Name	Description
	Nffv (⧉ see page 53)	Creates a new instance of the Nffv.

Nffv Methods

	Name	Description
	clearUsers (⧉ see page 53)	Removes all users from a database.
	contains (⧉ see page 53)	Checks if the database contains a concrete user.
	enroll (⧉ see page 53)	Gets a fingerprint from a scanner and saves it to a database.
	finalize (⧉ see page 53)	Implements standard Java method used by the garbage collector.
	getAvailableScannerModules (⧉ see page 54)	Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.
	getEngineStatus (⧉ see page 54)	Gets status information of the com.neurotechnology.Nffv (⧉ see page 51).
	getMatchingThreshold (⧉ see page 54)	Gets the minimum similarity value that verification function uses to determine whether the fingerprint matches.
	getMaxUserCount (⧉ see page 54)	The maximum number of users that can be enrolled to a database.
	getQualityThreshold (⧉ see page 54)	Gets image quality threshold.
	getUserByID (⧉ see page 54)	Returns a user details by the Id.
	getUsers (⧉ see page 55)	Gets a list of users enrolled to a database.
	removeUser (⧉ see page 55)	Removes a concrete user from a database.
	removeUserID (⧉ see page 55)	Removes user's ID.
	setEngineStatus (⧉ see page 55)	Sets status information for the com.neurotechnology.Nffv (⧉ see page 51).
	setIntEngineStatus (⧉ see page 55)	Sets status information for the com.neurotechnology.Nffv (⧉ see page 51).
	setMatchingThreshold (⧉ see page 56)	Sets the minimum similarity value that verification function uses to determine whether the fingerprint matches.
	setQualityThreshold (⧉ see page 56)	Sets image quality threshold.

◆	verify (see page 56)	Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

4.3.2.1.1.1 Nffv.Nffv Constructor

Creates a new instance of the Nffv.

Parameters

Parameters	Description
String database	Name of a database,
String password	Password for a database.
ScannerModule[] scannerModules	List of scanner modules that should be loaded.

Java

```java
public Nffv(String database, String password, ScannerModule[] scannerModules);
```

4.3.2.1.1.2 Nffv Methods

4.3.2.1.1.2.1 Nffv.clearUsers Method

Removes all users from a database.

Java

```java
public void clearUsers();
```

4.3.2.1.1.2.2 Nffv.contains Method

Checks if the database contains a concrete user.

Parameters

Parameters	Description
NffvUser user	User details to check.

Returns

Boolean value indicating if a database contains a concrete user.

Java

```java
public boolean contains(NffvUser user);
```

4.3.2.1.1.2.3 Nffv.enroll Method

Gets a fingerprint from a scanner and saves it to a database.

Parameters

Parameters	Description
int timeout	Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than timeout milliseconds.

Returns

NffvUser (see page 59) object that contains (see page 53) details of an enrolled user.

Java

```java
public NffvUser enroll(int timeout);
```

4.3.2.1.1.2.4 Nffv.finalize Method

Implements standard Java method used by the garbage collector.

Java

```
public void finalize() throws Throwable;
```

4.3.2.1.1.2.5 Nffv.getAvailableScannerModules Method

Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.

Returns

Array that contains (see page 53) available scanner modules.

Java

```
public static ScannerModule getAvailableScannerModules();
```

4.3.2.1.1.2.6 Nffv.getEngineStatus Method

Gets status information of the com.neurotechnology.Nffv (see page 51).

Returns

NffvStatus (see page 32) object that holds information about Nffv (see page 52).

Java

```
public NffvStatus getEngineStatus();
```

4.3.2.1.1.2.7 Nffv.getMatchingThreshold Method

Gets the minimum similarity value that verification function uses to determine whether the fingerprint matches.

Returns

The minimum similarity value that verification function accept for the same finger fingerprints. The default value is 0.01 %.

Java

```
public int getMatchingThreshold();
```

4.3.2.1.1.2.8 Nffv.getMaxUserCount Method

The maximum number of users that can be enrolled to a database.

Returns

The maximum number of users that can be enrolled to a database.

Java

```
public static native int getMaxUserCount();
```

4.3.2.1.1.2.9 Nffv.getQualityThreshold Method

Gets image quality threshold.

Returns

Returns fingerprint quality threshold. The value should be in range [0, 255]. The default value is 100.

Java

```
public int getQualityThreshold();
```

4.3.2.1.1.2.10 Nffv.getUserByID Method

Returns a user details by the Id.

Parameters

Parameters	Description
int id	User Id.

Returns

NffvUser (see page 59) object that contains (see page 53) user details.

Java

```
public NffvUser getUserByID(int id);
```

4.3.2.1.1.2.11 Nffv.getUsers Method

Gets a list of users enrolled to a database.

Returns

List of users that was enrolled to a database.

Java

```
public List<NffvUser> getUsers();
```

4.3.2.1.1.2.12 Nffv.removeUser Method

Removes a concrete user from a database.

Parameters

Parameters	Description
NffvUser user	NffvUser (see page 59) object that should be removed.

Java

```
public void removeUser(NffvUser user) throws Exception;
```

4.3.2.1.1.2.13 Nffv.removeUserID Method

Removes user's ID.

Parameters

Parameters	Description
int ID	User's ID to remove.

Java

```
public void removeUserID(int ID);
```

4.3.2.1.1.2.14 Nffv.setEngineStatus Method

Sets status information for the com.neurotechnology.Nffv (see page 51).

Parameters

Parameters	Description
NffvStatus engineStatus	NffvStatus (see page 32) object that holds information to set.

Java

```
protected void setEngineStatus(NffvStatus engineStatus);
```

4.3.2.1.1.2.15 Nffv.setIntEngineStatus Method

Sets status information for the com.neurotechnology.Nffv (see page 51).

Parameters

Parameters	Description
int engineStatus	Number that indicates Nffv (see page 52) status.

Java

```
protected void setIntEngineStatus(int engineStatus);
```

4.3.2.1.1.2.16 Nffv.setMatchingThreshold Method

Sets the minimum similarity value that verification function uses to determine whether the fingerprint matches.

Parameters

Parameters	Description
int value	The minimum similarity value that verification function accept for the same finger fingerprints.

Java

```
public void setMatchingThreshold(int value);
```

4.3.2.1.1.2.17 Nffv.setQualityThreshold Method

Sets image quality threshold.

Parameters

Parameters	Description
Image	quality threshold to set.

Java

```
public void setQualityThreshold(int value);
```

4.3.2.1.1.2.18 Nffv.verify Method

Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

Parameters

Parameters	Description
NffvUser user	A reference to a database record that should be matched with the scanned fingerprint.
int timeout	Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than timeout milliseconds.

Returns

Matching score.

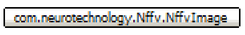
Java

```
public int verify(NffvUser user, int timeout);
```

4.3.2.1.2 NffvImage Class

Provides methods for managing images.

Class Hierarchy



Java

```
public class NffvImage;
```

NffvImage Methods

	Name	Description
◆	getBufferedImage (see page 57)	Gets a buffered image.
◆	getHeight (see page 57)	Retrieves image height.

	getHorizontalResolution (see page 57)	Gets the horizontal resolution of the image.
	getImageData (see page 57)	Gets an image data as a byte array.
	getImageIcon (see page 58)	Get an image icon.
	getStride (see page 58)	Gets the stride (size of one row) of the image.
	getVerticalResolution (see page 58)	Retrieves the vertical resolution of the image.
	getWidth (see page 58)	Retrieves image width.
	setHeight (see page 58)	Sets image height.
	setHorizontalResolution (see page 58)	Sets the horizontal resolution of the image.
	setImageData (see page 59)	Creates an image from a byte array.
	setStride (see page 59)	Sets the stride (size of one row) of the image.
	setVerticalResolution (see page 59)	Sets the vertical resolution of the image.
	setWidth (see page 59)	Sets image width.

4.3.2.1.2.1 NffvImage Methods

4.3.2.1.2.1.1 NffvImage.getBufferedImage Method

Gets a buffered image.

Returns

Buffered image which contains an accessible buffer of the image.

Java

```java
public BufferedImage getBufferedImage();
```

4.3.2.1.2.1.2 NffvImage.getHeight Method

Retrieves image height.

Returns

Image height in pixels.

Java

```java
public int getHeight();
```

4.3.2.1.2.1.3 NffvImage.getHorizontalResolution Method

Gets the horizontal resolution of the image.

Returns

Horizontal resolution of image.

Java

```java
public float getHorizontalResolution();
```

4.3.2.1.2.1.4 NffvImage.getImageData Method

Gets an image data as a byte array.

Returns

Byte array that contains image data.

Java

```java
public byte getImageData();
```

4.3.2.1.2.1.5 NffvImage.getImageIcon Method

Get an image icon.

Returns

An icon of the image.

Java

```
public ImageIcon getImageIcon();
```

4.3.2.1.2.1.6 NffvImage.getStride Method

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

Returns

Image stride.

Java

```
public int getStride();
```

4.3.2.1.2.1.7 NffvImage.getVerticalResolution Method

Retrieves the vertical resolution of the image.

Returns

Vertical resolution of image.

Java

```
public float getVerticalResolution();
```

4.3.2.1.2.1.8 NffvImage.getWidth Method

Retrieves image width.

Returns

Image width in pixels.

Java

```
public int getWidth();
```

4.3.2.1.2.1.9 NffvImage.setHeight Method

Sets image height.

Parameters

Parameters	Description
int height	Image height in pixels.

Java

```
public void setHeight(int height);
```

4.3.2.1.2.1.10 NffvImage.setHorizontalResolution Method

Sets the horizontal resolution of the image.

Parameters

Parameters	Description
float horizontalResolution	Horizontal resolution of image to set.

Java

```
public void setHorizontalResolution(float horizontalResolution);
```

4.3.2.1.2.1.11 NffvImage.setImageData Method

Creates an image from a byte array.

Parameters

Parameters	Description
byte [] imageData	Byte array that contains image data.

Java

```
public void setImageData(byte [] imageData);
```

4.3.2.1.2.1.12 NffvImage.setStride Method

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

Parameters

Parameters	Description
int stride	Image stride.

Java

```
public void setStride(int stride);
```

4.3.2.1.2.1.13 NffvImage.setVerticalResolution Method

Sets the vertical resolution of the image.

Parameters

Parameters	Description
float verticalResolution	Vertical resolution of image to set.

Java

```
public void setVerticalResolution(float verticalResolution);
```

4.3.2.1.2.1.14 NffvImage.setWidth Method

Sets image width.

Parameters

Parameters	Description
int width	Image width in pixels.

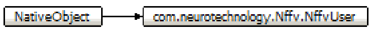
Java

```
public void setWidth(int width);
```

4.3.2.1.3 NffvUser Class

Provides methods for working with users.

Class Hierarchy



Java

```
public class NffvUser extends NativeObject;
```

NffvUser Methods

	Name	Description
◆	getID (↗ see page 60)	Retrieves from a database user's ID. If a user was disposed or removed from engine an error is thrown.
◆	getNffvImage (↗ see page 60)	Gets tan Image from the com.neurotechnology.Nffv (↗ see page 51).

◆	toString (see page 60)	Gets a string representation of object.

4.3.2.1.3.1 NffvUser Methods

4.3.2.1.3.1.1 NffvUser.getID Method

Retrieves from a database user's ID. If a user was disposed or removed from engine an error is thrown.

Returns

User's ID.

Java

```
public int getID() throws Exception;
```

4.3.2.1.3.1.2 NffvUser.getNffvImage Method

Gets tan Image from the com.neurotechnology.Nffv (see page 51).

Returns

NffvImage (see page 56) object.

Java

```
public NffvImage getNffvImage() throws Exception;
```

4.3.2.1.3.1.3 NffvUser.toString Method

Gets a string representation of object.

Returns

String representation of object.

Java

```
public String toString();
```

4.3.2.1.4 ScannerModule Class

Provides methods for setting and getting scanner names from the ScannerModule.

Class Hierarchy

Java

```
public class ScannerModule;
```

Methods

	Name	Description
◆	ScannerModule (see page 60)	Creates a new instance of the ScannerModule

ScannerModule Methods

	Name	Description
◆	getName (see page 61)	Gets a fingerprint scanner name.
◆	setName (see page 61)	Sets a name of a fingerprint scanner.

4.3.2.1.4.1 ScannerModule.ScannerModule Constructor

Creates a new instance of the ScannerModule

Parameters

Parameters	Description
String name	A name of a scanner to set.

Java

```
protected ScannerModule(String name);
```

4.3.2.1.4.2 ScannerModule Methods

4.3.2.1.4.2.1 ScannerModule.getName Method

Gets a fingerprint scanner name.

Returns

String that contains fingerprint scanner name.

Java

```
public String getName();
```

4.3.2.1.4.2.2 ScannerModule.setName Method

Sets a name of a fingerprint scanner.

Parameters

Parameters	Description
String name	A name of a scanner to set.

Java

```
protected void setName(String name);
```

4.4 Delphi Reference

This chapter provides the Free Fingerprint Verification SDK programming reference for Delphi programming language.

Notes

These files under directory *\samples\Delphi* are used for building Delphi wrapper of the FFV SDK:

- Nffv.pas
- NffvUser.pas

Also you can read Delphi sample chapter for more information.

Namespaces

Name	Description
Nffv (see page 61)	Contains classes and methods that provide the Free Fingerprint Verification SDK functionality.
NffvUser (see page 68)	Provides methods and properties for working with users.

4.4.1 Nffv Namespace

Contains classes and methods that provide the Free Fingerprint Verification SDK functionality.

Module

Delphi Reference (see page 61)

Classes

	Name	Description
	TNffv (see page 62)	The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with users and fingerprints.

Constants

Name	Description
dllName (see page 68)	Name of the dll that provides the main functionality of the FFV SDK.

Functions

	Name	Description
	EngineStatusString (see page 67)	Gets a string message that hold information about TNffv (see page 62) status.
	GetAvailableScannerModules (see page 67)	Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.
	NffvFreeMemory (see page 67)	Releases memory allocated by the GetAvailableScannerModules (see page 67) function.
	NffvGetInfo (see page 67)	Gets information about the Nffv library.

Structs, Records, Enums

	Name	Description
	TNffvStatus (see page 68)	Enumerates enrollment or verification status values.

4.4.1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	TNffv (see page 62)	The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with users and fingerprints.

4.4.1.1.1 TNffv Class

The main class of the Free Fingerprint Verification SDK. Provides methods and properties for working with users and fingerprints.

Pascal

```
TNffv = class;
```

Class Hierarchy

Methods

	Name	Description
	Create (see page 63)	Creates a new instance of the TNffv.
	Destroy (see page 63)	Releases resources used by object.

TNffv Methods

	Name	Description
	Cancel (see page 64)	Cancels a fingerprint enrollment or verification operation.
	Enroll (see page 64)	Gets a fingerprint from a scanner and saves it to a database.
	GetMatchingThreshold (see page 64)	Gets the minimum similarity value that verification function uses to determine whether the fingerprint matches.
	GetQualityThreshold (see page 64)	Gets image quality threshold.
	GetUserById (see page 64)	Returns a user details by Id.
	GetUserByIndex (see page 65)	Returns a user details by the index.
	GetUserCount (see page 65)	Gets the number of users that the Nffv (see page 61) contains.
	GetUserIndexById (see page 65)	Returns an index of the user specified by Id.
	RemoveUser (see page 65)	Removes a user specified by an index from a database.
	RemoveUsers (see page 66)	Removes all users from a database.
	SetMatchingThreshold (see page 66)	Sets the minimum similarity value that verification function uses to determine whether the fingerprint matches.
	SetQualityThreshold (see page 66)	Sets image quality threshold.
	Verify (see page 66)	Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

4.4.1.1.1.1 Create Constructor

4.4.1.1.1.1.1 TNffv.Create Constructor (string, string)

Creates a new instance of the TNffv (see page 62).

Pascal

```
constructor Create(databaseName: string; password: string); overload;
```

Parameters

Parameters	Description
databaseName: string	Database name. This database is used for storing user details and fingerprints.
password: string	Password for database.

4.4.1.1.1.1.2 TNffv.Create Constructor (string, string, string)

Creates a new instance of the TNffv (see page 62).

Pascal

```
constructor Create(databaseName: string; password: string; scannerModules: string);
overload;
```

Parameters

Parameters	Description
databaseName: string	Database name. This database is used for storing user details and fingerprints.
password: string	Password for database.
scannerModules: string	String that contains a list of scanners to load.

4.4.1.1.1.2 TNffv.Destroy Destructor

Releases resources used by object.

Pascal

```
destructor Destroy; override;
```

4.4.1.1.1.3 TNffv Methods

4.4.1.1.1.3.1 TNffv.Cancel Method

Cancels a fingerprint enrollment or verification operation.

Pascal

```
procedure Cancel;
```

Remarks

This method is useful when the fingerprint enrollment or verification operation take too long. In this case a message box can be shown for a user to cancel this operation.

4.4.1.1.1.3.2 TNffv.Enroll Method

Gets a fingerprint from a scanner and saves it to a database.

Pascal

```
function Enroll(timeout: LongWord; var engineStatus: TNffvStatus): TNffvUser;
```

Parameters

Parameters	Description
timeout: LongWord	Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than timeout milliseconds.
var engineStatus: TNffvStatus	The enrollment status value.

Returns

A reference to TNffvUser (see page 69) object which provides methods for managing enrolled users.

4.4.1.1.1.3.3 TNffv.GetMatchingThreshold Method

Gets the minimum similarity value that verification function uses to determine whether the fingerprint matches.

Pascal

```
function GetMatchingThreshold: LongInt;
```

Returns

The minimum similarity value that verification function accept for the same finger fingerprints. The default value is 0.01 %.

4.4.1.1.1.3.4 TNffv.GetQualityThreshold Method

Gets image quality threshold.

Pascal

```
function GetQualityThreshold: Byte;
```

Returns

The fingerprint quality threshold. The value should be in range [0, 255]. The default value is 100.

4.4.1.1.1.3.5 TNffv.GetUserById Method

Returns a user details by Id.

Pascal

```
function GetUserById(id: LongInt): TNffvUser;
```

Parameters

Parameters	Description
id: LongInt	User's identification number.

Returns

A reference to TNffvUser (see page 69) object which provides methods for managing enrolled users.

4.4.1.1.1.3.6 TNffv.GetUserByIndex Method

Returns a user details by the index.

Pascal

```
function GetUserByIndex(index: LongInt): TNffvUser;
```

Parameters

Parameters	Description
index: LongInt	User's index.

Returns

A reference to TNffvUser (see page 69) object which provides methods for managing enrolled users.

4.4.1.1.1.3.7 TNffv.GetUserCount Method

Gets the number of users that the Nffv (see page 61) contains.

Pascal

```
function GetUserCount: LongInt;
```

Returns

The number of users in the Nffv (see page 61).

4.4.1.1.1.3.8 TNffv.GetUserIndexById Method

Returns an index of the user specified by Id.

Pascal

```
function GetUserIndexById(id: LongInt): LongInt;
```

Parameters

Parameters	Description
id: LongInt	User's ID.

Returns

Index of the user specified by Id.

4.4.1.1.1.3.9 TNffv.RemoveUser Method

Removes a user specified by an index from a database.

Pascal

```
procedure RemoveUser(index: LongInt);
```

Parameters

Parameters	Description
index: LongInt	User's index.

4.4.1.1.1.3.10 TNffv.RemoveUsers Method

Removes all users from a database.

Pascal

```
procedure RemoveUsers;
```

4.4.1.1.1.3.11 TNffv.SetMatchingThreshold Method

Sets the minimum similarity value that verification function uses to determine whether the fingerprint matches.

Pascal

```
procedure SetMatchingThreshold(threshold: LongInt);
```

Parameters

Parameters	Description
threshold: LongInt	The minimum similarity value that verification function accept for the same finger fingerprints. The default value is 0.01 %.

4.4.1.1.1.3.12 TNffv.SetQualityThreshold Method

Sets image quality threshold.

Pascal

```
procedure SetQualityThreshold(threshold: Byte);
```

Parameters

Parameters	Description
threshold: Byte	The fingerprint quality threshold. The value should be in range [0, 255]. The default value is 100.

4.4.1.1.1.3.13 TNffv.Verify Method

Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

Pascal

```
function Verify(user: TNffvUser; timeout: LongWord; var engineStatus: TNffvStatus): LongInt;
```

Parameters

Parameters	Description
user: TNffvUser	A reference to a database record that should be matched with the scanned fingerprint.
timeout: LongWord	Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than timeout milliseconds.
var engineStatus: TNffvStatus	Verification status value.

Returns

This function returns a matching score.

4.4.1.2 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	EngineStatusString (see page 67)	Gets a string message that hold information about TNffv (see page 62) status.
	GetAvailableScannerModules (see page 67)	Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.
	NffvFreeMemory (see page 67)	Releases memory allocated by the GetAvailableScannerModules (see page 67) function.
	NffvGetInfo (see page 67)	Gets information about the Nffv (see page 61) library.

4.4.1.2.1 Nffv.EngineStatusString Function

Gets a string message that hold information about TNffv (see page 62) status.

Pascal

```
function EngineStatusString(status: TNffvStatus): string;
```

Parameters

Parameters	Description
status: TNffvStatus	NffvStatus (see page 32) object.

Returns

String message that hold information about TNffv (see page 62) status.

4.4.1.2.2 Nffv.GetAvailableScannerModules Function

Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.

Pascal

```
function GetAvailableScannerModules: String;
```

Returns

String that hold available fingerprint scanners. Each fingerprint scanner module is separated by a semicolon.

4.4.1.2.3 Nffv.NffvFreeMemory Function

Releases memory allocated by the GetAvailableScannerModules (see page 67) function.

Pascal

```
procedure NffvFreeMemory(point: PChar); stdcall;
```

Parameters

Parameters	Description
point: PChar	Memory block to release.

4.4.1.2.4 Nffv.NffvGetInfo Function

Gets information about the Nffv (see page 61) library.

Pascal

```
function NffvGetInfo: TNLibraryInfo;
```

Returns

Object which type is TNlibraryInfo.

4.4.1.3 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	TNffvStatus (see page 68)	Enumerates enrollment or verification status values.

4.4.1.3.1 Nffv.TNffvStatus Enumeration

Enumerates enrollment or verification status values.

Pascal

```
TNffvStatus = (
  nfesTemplateCreated = 1,
  nfesNoScanner = 2,
  nfesScannerTimeout = 3,
  nfesUserCanceled = 4,
  nfesQualityCheckFailed = 100
);
```

Members

Members	Description
nfesTemplateCreated = 1	Indicates that the fingerprint template was created.
nfesNoScanner = 2	Indicates that there is no fingerprint scanner connected.
nfesScannerTimeout = 3	Indicates that the fingerprint scanner has reached the timeout.
nfesUserCanceled = 4	Indicates that a user has canceled a fingerprint scanning.
nfesQualityCheckFailed = 100	Indicates that the Free Fingerprint Verification SDK had failed to check the quality of a fingerprint.

4

4.4.1.4 Constants

The following table lists constants in this documentation.

Constants

Name	Description
dllName (see page 68)	Name of the dll that provides the main functionality of the FFV SDK.

4.4.1.4.1 Nffv.dllName Constant

Name of the dll that provides the main functionality of the FFV SDK.

Pascal

```
dllName = 'Nffv.dll';
```

4.4.2 NffvUser Namespace

Provides methods and properties for working with users.

Module

Delphi Reference (see page 61)

Classes

	Name	Description
●	TNffvUser (see page 69)	Provides methods and properties for working with users.

Constants

Name	Description
dllName (see page 70)	Name of the dll that provides the main functionality of the FFV SDK.

4.4.2.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
●	TNffvUser (see page 69)	Provides methods and properties for working with users.

4.4.2.1.1 TNffvUser Class

Provides methods and properties for working with users.

Pascal

```
TNffvUser = class;
```

Class Hierarchy

NffvUser.TNffvUser

Methods

	Name	Description
◆	Create (see page 69)	Creates a new instance of the TNffvUser.

TNffvUser Methods

	Name	Description
◆	GetId (see page 69)	Retrieves user's Id.
◆	GetImage (see page 70)	Retrieves a fingerprint image of the concrete user.

4.4.2.1.1.1 TNffvUser.Create Constructor

Creates a new instance of the TNffvUser (see page 69).

Pascal

```
constructor Create(handle: Pointer);
```

4.4.2.1.1.2 TNffvUser Methods

4.4.2.1.1.2.1 TNffvUser.GetId Method

Retrieves user's Id.

Pascal

```
function GetId: LongInt;
```

Returns

User's Id.

4.4.2.1.1.2.2 TNffvUser.GetImage Method

Retrieves a fingerprint image of the concrete user.

Pascal

```
function GetImage: TBitmap;
```

Returns

Bitmap object that contains user's fingerprint data.

4.4.2.2 Constants

The following table lists constants in this documentation.

Constants

Name	Description
dllName (see page 70)	Name of the dll that provides the main functionality of the FFV SDK.

4.4.2.2.1 NffvUser.dllName Constant

Name of the dll that provides the main functionality of the FFV SDK.

Pascal

```
dllName = 'Nffv.dll';
```

4.5 VB6 Reference

This chapter provides the Free Fingerprint Verification SDK programming reference for VB6 programming language.

Notes

These files under directory *\samples\VB6* are used for building VB6 wrapper of the FFV SDK:

- Nffv.cls
- NffvUser.cls
- Nffv.bas

4.5.1 Functions

4.5.1.1 ClearUsers

Removes all users from a users database.

VB6

```
Public Sub ClearUsers()
```

4.5.1.2 Enroll

Gets a fingerprint from a scanner and saves it to a database.

Parameters

Parameters	Description
timeout	Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than timeout milliseconds.
engineUser	User that should be enrolled to database.

Returns

A reference to NffvUser (see page 75) object which provides methods for managing enrolled users.

VB6

```
Public Function Enroll(ByVal timeout As Long, ByRef engineUser As NffvUser) As NffvStatus
```

4.5.1.3 GetHandle

Gets a handle to the NffvUser (see page 75).

Returns

A handle to the NffvUser (see page 75).

VB6

```
Friend Function GetHandle() As Long
```

4.5.1.4 GetHBitmap

Gets a handle to the bitmap of a user fingerprint.

Returns

Handle to a bitmap object.

VB6

```
Friend Function GetHBitmap() As Long
```

4.5.1.5 GetImage

Gets a fingerprint image which was enrolled to a database for user.

Returns

IPictureDisp object that exposes the picture object's properties.

VB6

```
Friend Function GetImage() As IPictureDisp
```

4.5.1.6 GetMatchingThreshold

Gets the minimum similarity value that verification function uses to determine whether the fingerprint matches.

Returns

The minimum similarity value that verification function accept for the same finger fingerprints.

See Also

Matching Threshold (see page 15)

VB6

```
Public Function GetMatchingThreshold() As Long
```

4.5.1.7 GetQualityThreshold

Gets image quality threshold.

Returns

The fingerprint quality threshold. The value should be in range [0, 255]. The default value is 100.

See Also

Quality Threshold (see page 15)

VB6

```
Public Function GetQualityThreshold() As Byte
```

4.5.1.8 GetUser

Returns a user details by the specified index.

Parameters

Parameters	Description
index	Index of a user in the database.

Returns

A reference to NffvUser (see page 75) object which provides methods for managing enrolled users.

VB6

```
Public Function GetUser(ByVal index As Long) As NffvUser
```

4.5.1.9 GetUserCount

Gets the number of users that the Nffv (see page 74) contains.

Returns

The number of users in the Nffv (see page 74).

VB6

```
Public Function GetUserCount() As Long
```

4.5.1.10 GetUserId

Returns user's Id.

Returns

Id that identifies a user.

VB6

```
Friend Function GetUserId() As Long
```

4.5.1.11 Nffv_GetAvailableScannerModules

Returns available fingerprint scanner modules for usage in the Free Fingerprint Verification SDK.

Returns

A string that contains the list of scanners separated by semicolons.

VB6

```
Public Function Nffv_GetAvailableScannerModules() As String
```

4.5.1.12 NLibraryInfo

This type provides library information.

Parameters

Parameters	Description
Title	Title of the library.
Product	Product's name.
Company	Company's name.
Copyright	Copyright notice from the library.
VersionMajor	Library's major version.
VersionMinor	Library's minor version.
VersionBuild	Library build version.
VersionRevision	Library's revision number.
DistributorId	Library's Id.
SerialNumber	Library's serial number.

VB6

```
Public Type NLibraryInfo Title As String * 64 Product As String * 64 Company As String * 64
Copyright As String * 64 VersionMajor As Long VersionMinor As Long VersionBuild As Long
VersionRevision As Long DistributorId As Long SerialNumber As Long End Type
```

4.5.1.13 RemoveUser

Removes a user specified by an index from a database.

Parameters

Parameters	Description
index	Index of a user that should be removed.

VB6

```
Public Sub RemoveUser(ByVal index As Long)
```

4.5.1.14 SetMatchingThreshold

Sets the minimum similarity value that verification function uses to determine whether the fingerprint matches.

Parameters

Parameters	Description
matchingThreshold	Matching threshold to set.

See Also

Matching Threshold (see page 15)

VB6

```
Public Sub SetMatchingThreshold(ByVal matchingThreshold As Long)
```

4.5.1.15 SetQualityThreshold

Sets image quality threshold.

Parameters

Parameters	Description
qualityThreshold	Quality threshold to set.

See Also

Quality Threshold (see page 15)

VB6

```
Public Sub SetQualityThreshold(ByVal qualityThreshold As Byte)
```

4.5.1.16 Verify

Compares a captured fingerprint with the one that was enrolled to a database before in order to determine whether two match.

Parameters

Parameters	Description
engineUser	Selected user that should be verified.
timeout	Specifies the time in milliseconds after which the fingerprint scanner stops scanning fingerprint. This usually happens when a finger is removed from a scanner for longer than timeout milliseconds.
score	Matching score of verification.

Returns

This function returns a reference to the NffvStatus (see page 32).

VB6

```
Public Function Verify(ByRef engineUser As NffvUser, ByVal timeout As Long, ByRef score As
Long) As NffvStatus
```

4.5.2 Types

4.5.2.1 Nffv

This type provides the main functionality of the FFV SDK.

4.5.2.2 NffvStatus

Enumerates different enrollment and verification status values.

Parameters

Parameters	Description
nfesTemplateCreated	Indicates that the fingerprint template was created.
nfesNoScanner	Indicates that there is no fingerprint scanner connected.
nfesScannerTimeout	Indicates that the fingerprint scanner has reached the timeout.
nfesQualityCheckFailed	Indicates that the Free Fingerprint Verification SDK had failed to check the quality of a fingerprint.

VB6

```
Public Enum NffvStatus nfesNone = 0 nfesTemplateCreated = 1 nfesNoScanner = 2
nfesScannerTimeout = 3 nfesQualityCheckFailed = 100 End Enum
```

4.5.2.3 NffvUser

This type is used to define a user who was enrolled to database.

5 Distribution Content

The Free Fingerprint Verification SDK contains these files and folders:

/bin

/Win32_x86 - this directory contains files for Windows-based operating systems running on 32-bit x86 CPU. This directory contains these files:

File	Description
CppSample.exe	An executable C++ sample application.
CSharpSample.exe	An executable C# sample application.
DelphiSample.exe	An executable Delphi sample application.
Neurotec.Biometrics.Nffv.dll	Provides the main functionality of the FFV SDK. Required Dll is Nffv.dll
Nffv.dll	Provides the main functionality of the FFV SDK.
VBNETSample.exe	An executable VB.NET sample application.
NffvJavaNative.dll NffvSample.html NffvSample.jar	Files for Java sample.
NffvServer.exe	Nffv.dll helper.

/Win32_x86/FPSmm - a directory containing scanner files:

File	Description
FPSmmJstac.dll WIS_API.dll WisCmos2.dll	Athena 210 module.
Additional/FPSmmIdentix.dll	Identix DFR 2080, DFR 2090, DFR 2100 scanners module. Drivers not included. FPSmmIdentix.dll file used with Itf32_2080U2.dll.
Additional/FPSmmNitgen.dll	NITGEN Fingkey Hamster & Fingkey Hamster II scanners module. Drivers included: "\install\FingerprintScanners\NITGEN\". FPSmmNitgen.dll file used with NBioBSP.dll.
Additional/FPSmmSecugenHFDU02.dll Additional/FPSmmSecugenHFDU03.dll Additional/FPSmmSecugenHFDU04.dll	SecuGen Hamster III, Hamster Plus, Hamster IV modules. Drivers included: "\install\Fingerprint Scanners\SecuGen\".
FPSmmAuthentec.dll	AuthenTec AES4000, AF-S2 sensors module.
Additional/FPSmmAuthenTec2501.dll ATSC63.dll	AuthenTec AES2501B module.
FPSmmAtmel.dll FingerChip.dll	Atmel FingerChip sensor module. Drivers included: "\install\Fingerprint Scanners\Atmel\".
FPSmmBiometrika.dll fx3.dll fx3scan.dll	Biometrika FX2000, FX3000 and HiScan scanners module. Drivers included: "\install\Fingerprint Scanners\BiometriKa\".
FPSmmCertis.dll CertisExports.dll Id3BiokeyDll.dll	Certis Image scanner module. Drivers included: "\install\Fingerprint Scanners\id3\".

FPSmmCrossMatch.dll	CrossMatch V300/V300LC/LC2.0 scanners module. Drivers can be downloaded from CrossMatch website (option "USB SDK for Verifier and MV5 Scanners/Readers").
FPSmmCyte.dll CaptureSDK.dll	Testech Bio-i and LES scanner module. Drivers included: "\install\Fingerprint Scanners\Testech\".
FPSmmDactyScan.dll FSM26U.dll	Green Bit DactyScan 26 scanner module. Drivers not included.
FPSmmDigent.dll IZZIX.dll	Digent Izzix FD1000 scanner module. Drivers included: "\install\Fingerprint Scanners\Digent\".
FPSmmEikon.dll. TCI.dll	Module for UPEK Eikon, UPEK TouchChip TCRU2C, UPEK TouchChip TCRU1C fingerprint scanners.
FPSmmFM200.dll fm200api.dll FingerPrinter.dll fm200drv.dll	Startek FM200 scanner module. Drivers included: "\install\Fingerprint Scanners\Startek\".
FPSmmFujitsu.dll libusb0.dll	Fujitsu MBF200 scanner module. Drivers included: "\install\Fingerprint Scanners\Fujitsu\".
FPSmmFutronicEthernetFam.dll FPSmmFutronicEthernetFam.ini	Futronic Ethernet FAM scanner module. Remarks FPSmmFutronicEthernetFam.ini file is intended for scanner configuration. Scanner IP address and port should be placed in file. Drivers included: "\install\Fingerprint Scanners\Futronic\"
FPSmmFutronic.dll ftrScanAPI.dll	Futronic FS80, FS88, BioLink U-Match MatchBook v.3.5 scanners module. Drivers included: "\install\Fingerprint Scanners\Futronic\". Note Configuration "futronic.cfg" file with parameter LFD = false will turn of the life fingerprint detection. For BioLink U-Match MatchBook v.3.5 scanner file should be created all the time.
Itf32_2080U2.dll	Identix DFR 2080, DFR 2090, DFR 2100 scanners module. Drivers not included. Itf32_2080U2.dll file used with FPSmmIdentix.dll.
FPSmmLighTunning.dll GetImageC500.dll	LighTunning LTT-C500 scanner module. Drivers not included.
FPSmmLScanEssentials.dll	CrossMatch L Scan Essentials fingerprints scanner module.
FPSmmRealScan.dll	Suprema RealScan-D fingerprints scanner module.
FPSmmSOP1.dll sop1.dll	Module for SOP1 fingerprints scanner. sop1.dll file is used with FPSmmSOP1.dll.
FPSmmSuprema.dll	Suprema BioMini, Suprema SFR300-S fingerprints scanner module.
FPSmmUPEK.dll TCI.dll	UPEK TouchChip TCRU1C and TCRU2C sensors module. Drivers included: "\install\Fingerprint Scanners\UPEK\".
FPSmmVistaMT.dll VciMt.dll	Vista MT fingerprints scanner module. VciMt.dll file is used together with FPSmmVistaMT.dll.
FPSmmZKSensor6000	ZKSoftware ZK6000 fingerprints scanner module.
FPSmmTacoma.dll SmzCmos1.dll SMZ_API.dll	Tacoma CMOS scanner module. Drivers included: "\install\Fingerprint Scanners\Tacoma\".
FPSmmUareU.dll	Digital Persona U.are.U 2000/4000S/4000B scanner module. Drivers included: "\install\Fingerprint Scanners\DigitalPersona\".
NBioBSP.dll	NITGEN Fingkey Hamster & Fingkey Hamster II, NITGEN eNBioScan-F scanners module. Drivers included: "\install\Fingerprint Scanners\NITGEN\". NBioBSP.dll file used with FPSmmNitgen.dll.

LumiAPI.dll LumiCore.dll FPSmmLumidigm.dll	Lumidigm Venus Series sensor module. Drivers included: "\install\Fingerprint Scanners\Lumidigm\".
Additional\FPSmmTSTBIRD.dll TSTBasic.dll	TST Biometrics BiRD 3 scanner module. Drivers can be downloaded from TST Biometrics website http://www.tst-biometrics.com/.
FPSmmHongda.dll	Hongda S680 module. Drivers included: "\install\Fingerprint Scanners\Hongda\".
Additional\FPSmmDermalog.dll DermalogVC3.dll Other files DermalogCalibrateSDK.dll DermalogLoggingFacility.dll DermalogPLS1.cfg DermalogPLS1.dll ZFScanAPI.dll should be copied in current directory, e.i. in the same as FPScannerMan.dll.	Dermalog ZF1 module. Drivers included: "\install\Fingerprint Scanners\Dermalog\".
FPSmmDakty.dll DaktyImage.dll fpd.dll fpdusb.dll Segmentation.dll	Dakty Fingerprint NAOS-A module. Drivers included: "\install\Fingerprint Scanners\DaktyFpdNaosA\".

Recommendations:

1. "FPSmm" folder must be located in the same folder as Nffv.dll; "FPSmm" folder must contain required scanners modules.

2. Copy only required DLL files; for example, if CrossMatch is not used, FPSmmCrossMatch.dll should not be copied.

Known conflicts/issues:

1. Nitgen, Identix and SecuGen drivers can conflict with each other. We would recommend to use only one FPSmmIdentix.dll, FPSmmNitgen.dll or FPSmmSecuGen.dll file at the same time.

2. FPSmmCrossMatch.dll loading time is quite long; if CrossMatch scanner is not used, we would recommend to exclude FPSmmCrossMatch.dll from application distribution.

3. Dermalog ZF1 and Futronic FS80, FS88 scanners can not work together.

/documentation

File	Description
Free Fingerprint Verification SDK.pdf	Documentation file in pdf
Free Fingerprint Verification SDK.chm	Documentation file in chm
license.html	License file

/Include

/Windows

This directory contains header files.

/install/Fingerprint Scanners

This directory contains drivers for some fingerprint scanners.

/lib

/Win32_x86

This directory contains lib files that are used by the Free Fingerprint Verification SDK.

/redistributable

/Win32_x86

Files	Description
FFVSDKRedistributable.exe	A program used to redistribute your application (including Neurotechnology DLL files).
FFVSDKRedistributable.txt	Text file that describes how to use FFV SDK redistributable.

/samples

This directory contains files for sample applications. The folder under this directory:

Folder	Description
Cpp	This folder contains files for C++ sample application.
CSharp	This folder contains files for C# sample application.
Java	This folder contains files for Java sample application.
VB.NET	This folder contains files for VB.NET sample application.
VB6	This folder contains files for VB6 sample application.
Delphi	This folder contains files for Delphi sample application.

6 Error Codes

In this chapter the possible error codes are listed:

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

7 FAQ

There are listed frequently asked questions (FAQ) and answers to them.

1. **When I have enrolled 10 users to a database and try to add more users I get an error message "Nffv (see page 61) already contains NF_ENGINE_MAX_USER_COUNT users. Code: -7". How can I add more users?**

The Free Fingerprint Verification SDK allows to add up to 10 users to a database. If you need to add new users, you should remove other users.

If you have an intention of developing a larger scale application or system with unlimited users count, server or cluster support, please contact Neurotechnology for guidelines for other products.

2. **Is it possible to save an original fingerprint image to a hard disk?**

No, it is not possible. If you need to save an original fingerprint image, you should buy the VeriFinger SDK.

3. **Can I use several different fingerprint scanners?**

Yes, you can use different fingerprint scanners simultaneously.

4. **What is a maximum matching score of two fingerprints?**

There is no maximum score.

Index