



MegaMatcher On Card 2.0 SDK

Developer's Guide

Version: 2.0.0.0

Release date: 2010-08-06

Table of Contents

1 About 1

- 1.1 Supported Platforms 1
- 1.2 System Requirements 1
- 1.3 MegaMatcher OnCard Structure 2
- 1.4 Licensing 2
 - 1.4.1 Single computer license 3

2 Overview 4

- 2.1 On-Card Biometric Components 4
- 2.2 PC Part 5

3 Using 6

- 3.1 Java Card Part 6
 - 3.1.1 On-card Biometric APIs 6
 - 3.1.1.1 Using On-Card Bio API Library 7
 - 3.1.1.2 Using On-card Bio API Sample Applets 7
 - 3.1.2 JC Bio API Matching On-Card Library Configurations 8
- 3.2 PC Part 9

4 Tutorials 10

5 Samples 11

- 5.1 Smartcard Fingers Client 11
- 5.2 Smartcard Faces Client 11
- 5.3 Smartcard CAP Loader 12

6 API Reference 13

- 6.1 C/C++ Reference 13
 - 6.1.1 NBiometricStandards Library 13
 - 6.1.1.1 Bdif Module 14
 - Files 14
 - Bdif.h 14
 - 6.1.1.2 BdifTypes Module 14
 - Structs, Records, Enums 15
 - BdifEyeColor Enumeration 15
 - BdifEyePosition Enumeration 17

BdifFaceExpression Enumeration	17
BdifFaceFeaturePoint Structure	18
BdifFaceFeaturePointType Enumeration	19
BdifFaceProperties Enumeration	19
BdifFPatternClass Enumeration	21
BdifFPImpressionType Enumeration	22
BdifFPMinutiaNeighbor Structure	25
BdifFPMinutiaType Enumeration	25
BdifFPPosition Enumeration	26
BdifGender Enumeration	28
BdifHairColor Enumeration	29
BdifImageSourceType Enumeration	31
BdifIrisOrientation Enumeration	32
BdifIrisScanType Enumeration	33
BdifScaleUnits Enumeration	34
BdifStandard Enumeration	35
Macros	35
BDIF_DO_NOT_CHECK_CBEFF_PRODUCT_ID Macro	35
Files	36
BdifTypes.h	36
6.1.1.3 CbeffBdbFormatIdentifiers Module	37
Macros	40
CBEFF_BDBFI_ID3_SEMICONDUCTORS_COMPACT_CARD_EXTENSION Macro	41
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FACE_IMAGE Macro	41
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_IMAGE Macro	41
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_N Macro	41
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_V Macro	41
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_N Macro	42
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_V Macro	42
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_N Macro	42
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_X Macro	42
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_DISCRETE_FOURIER_TRANSFORM Macro	43
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_GABOR_FILTER Macro	43
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_QUANTIZED_CO_SINUSOIDAL_TRIPLET Macro	43
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_POLAR Macro	43
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_RECTILINEAR Macro	44
CBEFF_BDBFI_NEUROTECHNOLOGIJA_NLRECORD_3 Macro	44
CBEFF_BDBFI_NEUROTECHNOLOGIJA_NTTEMPLATE Macro	44
Files	44
CbeffBdbFormatIdentifiers.h	44
6.1.1.4 CbeffBiometricOrganizations Module	48

Macros 50

CBEFF_BO_123ID Macro 51
CBEFF_BO_A_4_VISION Macro 51
CBEFF_BO_ACSYS_BIOMETRICS Macro 51
CBEFF_BO_ACTIVCARD Macro 51
CBEFF_BO_ANTHEUS_TECHNOLOGY Macro 51
CBEFF_BO_ATMEL Macro 51
CBEFF_BO_AURORA_WIRELESS_TECHNOLOGIES Macro 52
CBEFF_BO_AUTHEN_TEC Macro 52
CBEFF_BO_AWARE Macro 52
CBEFF_BO_BIO_WISE Macro 52
CBEFF_BO_BIOKEY_INTERNATIONAL Macro 52
CBEFF_BO_BIOKEY_INTERNATIONAL_2 Macro 52
CBEFF_BO_BIOLINK_TECHNOLOGIES_INTERNATIONAL Macro 53
CBEFF_BO_BIOSCRYPT Macro 53
CBEFF_BO_BIOVISION Macro 53
CBEFF_BO_CYBER_SIGN Macro 53
CBEFF_BO_COGENT_SYSTEMS Macro 53
CBEFF_BO_CRYPTO_METRICS Macro 54
CBEFF_BO_CROSS_MATCH_TECHNOLOGIES Macro 54
CBEFF_BO_DERMALOG_IDENTIFICATION_SYSTEMS Macro 54
CBEFF_BO_DIGITALPERSONA Macro 54
CBEFF_BO_EAST_SHORE_TECHNOLOGIES Macro 54
CBEFF_BO_ECRYP Macro 54
CBEFF_BO_FINGERPRINT_CARDS Macro 55
CBEFF_BO_GERMAN_INSTITUTE_FOR_STANDARDIZATION Macro 55
CBEFF_BO_GREEN_BIT_AMERICAS Macro 55
CBEFF_BO_GRIAULE_TECNOLOGIA Macro 55
CBEFF_BO_GUARDWARE_SYSTEMS Macro 55
CBEFF_BO_HITACHI Macro 55
CBEFF_BO_HUMANSCAN Macro 56
CBEFF_BO_ID3_SEMICONDUCTORS Macro 56
CBEFF_BO_IDENTIX Macro 56
CBEFF_BO_IDENTIX_2 Macro 56
CBEFF_BO_IMAGE_WARE_SYSTEMS Macro 56
CBEFF_BO_INCITS_TC_M1_BIOMETRICS Macro 57
CBEFF_BO_INFINEON_TECHNOLOGIES Macro 57
CBEFF_BO_INNOVATRICS Macro 57
CBEFF_BO_INTERNATIONAL_BIOMETRIC_GROUP Macro 57
CBEFF_BO_INVESTORSHELPERS Macro 57
CBEFF_BO_IRIDIAN_TECHNOLOGIES Macro 57
CBEFF_BO_ISO_IEC_JTC_1_SC_37_BIOMETRICS Macro 58
CBEFF_BO_JANUS_ASSOCIATES Macro 58
CBEFF_BO_LG_ELECTRONICS_USA Macro 58

CBEFF_BO_LOGICO_SMARTCARD_SOLUTIONS Macro	58
CBEFF_BO_LUMIDIGM Macro	58
CBEFF_BO_NEC_SOLUTIONS_AMERICA Macro	58
CBEFF_BO_NEUROTECHNOLOGIJA Macro	59
CBEFF_BO_NIST Macro	59
CBEFF_BO_NITGEN Macro	59
CBEFF_BO_NOT_FOR_USE Macro	59
CBEFF_BO_OMNIPERCEPTION Macro	59
CBEFF_BO_PRECISE_BIOMETRICS Macro	60
CBEFF_BO_PRIVATE Macro	60
CBEFF_BO_RECOGNITION_SYSTEMS Macro	60
CBEFF_BO_SAFLINK Macro	60
CBEFF_BO_SAGEM_MORPHO Macro	60
CBEFF_BO_SECUGEN Macro	60
CBEFF_BO_SECURE_DESIGN Macro	61
CBEFF_BO_SILEX_TECHNOLOGY Macro	61
CBEFF_BO_SONDA_TECHNOLOGIES Macro	61
CBEFF_BO_STARTEK_ENGINEERING Macro	61
CBEFF_BO_SUPREMA Macro	61
CBEFF_BO_TESTING_1 Macro	61
CBEFF_BO_TESTING_2 Macro	62
CBEFF_BO_THALES_IDENTIFICATION Macro	62
CBEFF_BO_ULTRA_SCAN Macro	62
CBEFF_BO_UPEK Macro	62
CBEFF_BO_VALIDITY Macro	62
CBEFF_BO_VERIDICOM Macro	63
CBEFF_BO_VERIDT Macro	63
CBEFF_BO_VIISAGE Macro	63
CBEFF_BO_XTEC Macro	63
Files	63
CbeffBiometricOrganizations.h	63
6.1.1.5 FMCARD Module	66
Functions	67
FMCARDWriteCompactSizeMinutiae Function	67
FMCARDWriteNormalSizeMinutiae Function	68
Structs, Records, Enums	69
FmcFeatureHandling Enumeration	69
FmcMinutiaOrder Enumeration	69
Macros	70
FMC_BDT_TAG_CELL_QUALITY_DATA Macro	70
FMC_BDT_TAG_CORE_POINT_DATA Macro	70
FMC_BDT_TAG_DELTA_POINT_DATA Macro	71
FMC_BDT_TAG_FINGER_MINUTIAE_DATA Macro	71
FMC_BDT_TAG_RIDGE_COUNT_DATA Macro	71

FMC_BIT_BHT_BAPT_TAG_FEATURE_HANDLING_INDICATOR Macro	71
FMC_BIT_BHT_BAPT_TAG_MINUTIAE_ORDER Macro	71
FMC_BIT_BHT_BAPT_TAG_NUMBER_OF_MINUTIAE Macro	71
FMC_DEFAULT_MAX_ENROLL_MC Macro	72
FMC_DEFAULT_MAX_VERIFY_MC Macro	72
FMC_DEFAULT_MIN_ENROLL_MC Macro	72
FMC_DEFAULT_MIN_VERIFY_MC Macro	72
Files	72
FMC.h	72
6.1.1.6 NBiometricStandards Module	73
Functions	74
NBiometricStandardsGetInfo Function	74
Files	74
NBiometricStandards.h	74
6.1.2 NCore Library	74
6.1.2.1 NCore Module	75
Functions	75
NCoreGetInfo Function	75
NCoreOnExit Function	76
NCoreOnStart Function	76
NCoreOnThreadExit Function	76
NCoreOnThreadStart Function	76
Files	77
NCore.h	77
6.1.2.2 NErrors Module	77
Macros	78
N_E_ARGUMENT Macro	78
N_E_ARGUMENT_NULL Macro	78
N_E_ARGUMENT_OUT_OF_RANGE Macro	78
N_E_CLR Macro	79
N_E_COM Macro	79
N_E_CORE Macro	79
N_E_END_OF_STREAM Macro	79
N_E_EXTERNAL Macro	79
N_E_FAILED Macro	79
N_E_FORMAT Macro	80
N_E_INDEX_OUT_OF_RANGE Macro	80
N_E_INVALID_OPERATION Macro	80
N_E_IO Macro	80
N_E_NOT_ACTIVATED Macro	80
N_E_NOT_IMPLEMENTED Macro	81
N_E_NOT_SUPPORTED Macro	81
N_E_NULL_REFERENCE Macro	81
N_E_OUT_OF_MEMORY Macro	81

N_E_OVERFLOW Macro	81
N_E_PARAMETER Macro	81
N_E_PARAMETER_READ_ONLY Macro	82
N_E_SYS Macro	82
N_E_WIN32 Macro	82
N_OK Macro	82
NFailed Macro	82
NSucceeded Macro	83
Files	83
NErrors.h	83
6.1.2.3 NGeometry Module	84
Structs, Records, Enums	84
NPoint Structure	84
NRect Structure	85
NSize Structure	85
NPointD Structure	85
NPointF Structure	86
NRectD Structure	86
NRectF Structure	86
NSizeD Structure	87
NSizeF Structure	87
Files	87
NGeometry.h	87
6.1.2.4 NLibraryInfo Module	88
Structs, Records, Enums	88
NLibraryInfo_ Structure	89
Macros	89
N_LI_ACTIVATED_MAX_LENGTH Macro	89
N_LI_COMPANY_MAX_LENGTH Macro	89
N_LI_COPYRIGHT_MAX_LENGTH Macro	90
N_LI_PRODUCT_MAX_LENGTH Macro	90
N_LI_TITLE_MAX_LENGTH Macro	90
Files	90
NLibraryInfo.h	90
6.1.2.5 NMemory Module	91
Functions	91
NAlignedFree Function	91
NAlloc Function	91
NCAlloc Function	92
NCompare Function	92
NCopy Function	93
NFill Function	93
NFree Function	94
NMove Function	94

NReAlloc Function	95
Macros	95
NClear Macro	95
Files	95
NMemory.h	95
6.1.2.6 NObject Module	96
Functions	96
NObjectCopyParameters Function	96
NObjectFree Function	97
NObjectGetOwner Function	97
NObjectGetParameter Function	97
NObjectGetParameterWithPart Function	98
NObjectGetType Function	98
NObjectReset Function	99
NObjectSetParameter Function	99
NObjectSetParameterWithPart Function	99
NObjectTypeOf Function	100
NTypeGetBaseType Function	100
NTypeGetName Function	101
Files	101
NObject.h	101
6.1.2.7 NParameters Module	101
Macros	102
N_TYPE_BOOL Macro	102
N_PC_TYPE_ID Macro	102
N_TYPE_BYTE Macro	102
N_TYPE_CHAR Macro	103
N_TYPE_DOUBLE Macro	103
N_TYPE_FLOAT Macro	103
N_TYPE_INT Macro	103
N_TYPE_LONG Macro	103
N_TYPE_SBYTE Macro	104
N_TYPE_SHORT Macro	104
N_TYPE_STRING Macro	104
N_TYPE_UINT Macro	104
N_TYPE_ULONG Macro	104
N_TYPE_USHORT Macro	104
NParameterMakeld Macro	105
Files	105
NParameters.h	105
6.1.2.8 NProcessorInfo Module	106
Functions	106
NProcessorInfoGetModelNameA Function	106
NProcessorInfoGetModelNameW Function	107

NProcessorInfoGetVendor Function	107
NProcessorInfoGetVendorNameA Function	108
NProcessorInfoGetVendorNameW Function	108
NProcessorInfoIs3DNowSupported Function	109
NProcessorInfoIsMmxSupported Function	109
NProcessorInfoIsSse2Supported Function	109
NProcessorInfoIsSse3Supported Function	110
NProcessorInfoIsSseSupported Function	110
Structs, Records, Enums	110
NProcessorVendor Enumeration	110
Macros	111
N_PROCESSOR_INFO_MAX_MODEL_NAME_LENGTH Macro	111
N_PROCESSOR_INFO_MAX_VENDOR_NAME_LENGTH Macro	111
Files	111
NProcessorInfo.h	111
6.1.2.9 NTypes Module	112
Structs, Records, Enums	115
NIndexPair Structure	115
NRational Structure	115
NURational Structure	115
Types	116
NChar Type	116
NByte Type	116
NBool Type	116
NBoolean Type	116
NChar Type	117
NDouble Type	117
NFloat Type	117
NHandle Type	117
NInt Type	117
NInt16 Type	117
NInt32 Type	118
NInt64 Type	118
NInt8 Type	118
NLong Type	118
NPosType Type	118
NResult Type	118
NSByte Type	119
NShort Type	119
NSingle Type	119
NSizeType Type	119
NUInt Type	119
NUInt16 Type	120
NUInt32 Type	120

NUInt64 Type 120
NUInt8 Type 120
NULong Type 120
NUShort Type 120
NWChar Type 121
Macros 121
 N_64 Macro 121
 N_DECLARE_HANDLE Macro 121
 N_ANSI_C Macro 121
 N_BIG_ENDIAN Macro 121
 N_BYTE_MAX Macro 122
 N_BYTE_MIN Macro 122
 N_CALLBACK Macro 122
 N_CALLBACK_AW Macro 122
 N_CPP Macro 122
 N_DEBUG Macro 122
 N_DOUBLE_EPSILON Macro 123
 N_DOUBLE_MAX Macro 123
 N_DOUBLE_MIN Macro 123
 N_FAST_FLOAT Macro 123
 N_FLOAT_EPSILON Macro 123
 N_FLOAT_MAX Macro 123
 N_FLOAT_MIN Macro 124
 N_FUNC_AW Macro 124
 N_GCC Macro 124
 N_INT_MAX Macro 124
 N_INT_MIN Macro 124
 N_INT16_MAX Macro 125
 N_INT16_MIN Macro 125
 N_INT32_MAX Macro 125
 N_INT32_MIN Macro 125
 N_INT64_MAX Macro 125
 N_INT64_MIN Macro 125
 N_INT8_MAX Macro 126
 N_INT8_MIN Macro 126
 N_LIB Macro 126
 N_LINUX Macro 126
 N_LONG_MAX Macro 126
 N_LONG_MIN Macro 126
 N_MAC Macro 127
 N_MSVC Macro 127
 N_NO_ANSI_FUNC Macro 127
 N_NO_INT_64 Macro 127
 N_PACKED Macro 127

N_POS_TYPE_MAX Macro	128
N_POS_TYPE_MIN Macro	128
N_SBYTE_MAX Macro	128
N_SBYTE_MIN Macro	128
N_SHORT_MAX Macro	128
N_SHORT_MIN Macro	128
N_SINGLE_EPSILON Macro	129
N_SINGLE_MAX Macro	129
N_SINGLE_MIN Macro	129
N_SIZE_TYPE_MAX Macro	129
N_SIZE_TYPE_MIN Macro	129
N_STRUCT_AW Macro	129
N_T Macro	130
N_UINT_MAX Macro	130
N_UINT_MIN Macro	130
N_UINT16_MAX Macro	130
N_UINT16_MIN Macro	130
N_UINT32_MAX Macro	131
N_UINT32_MIN Macro	131
N_UINT64_MAX Macro	131
N_UINT64_MIN Macro	131
N_UINT8_MAX Macro	131
N_UINT8_MIN Macro	131
N_ULONG_MAX Macro	132
N_ULONG_MIN Macro	132
N_UNICODE Macro	132
N_USHORT_MAX Macro	132
N_USHORT_MIN Macro	132
N_WCHAR_SIZE Macro	132
N_WINDOWS Macro	133
NFalse Macro	133
NTrue Macro	133
NULL Macro	133
Files	133
NTypes.h	133
6.1.3 NDeviceManager Library	136
6.1.3.1 FPScanner Module	137
Functions	137
FPScannerGetIdA Function	137
FPScannerGetIdW Function	138
FPScannerIsCapturing Function	138
FPScannerSetFingerPlacedCallback Function	139
FPScannerSetFingerRemovedCallback Function	139
FPScannerSetImageScannedCallback Function	140

FPScannerSetIsCapturingChangedCallback Function	140
FPScannerStartCapturing Function	140
FPScannerStartCapturingForOneImage Function	141
FPScannerStopCapturing Function	141
Variables	142
hImage Variable	142
Files	142
FPScanner.h	142
6.1.3.2 FPScannerMan Module	143
Functions	143
FPScannerManGetAvailableModulesA Function	144
FPScannerManGetAvailableModulesW Function	144
FPScannerManGetScanner Function	144
FPScannerManGetScannerByIdA Function	145
FPScannerManGetScannerByIdW Function	145
FPScannerManGetScannerCount Function	146
FPScannerManInitialize Function	146
FPScannerManInitializeWithModulesA Function	147
FPScannerManInitializeWithModulesW Function	147
FPScannerManSetScannerAddedCallbackA Function	148
FPScannerManSetScannerAddedCallbackW Function	148
FPScannerManSetScannerRemovedCallbackA Function	149
FPScannerManSetScannerRemovedCallbackW Function	149
FPScannerManUninitialize Function	149
Variables	150
hScanner Variable	150
void * pParam) Variable	150
Files	150
FPScannerMan.h	150
6.1.3.3 NDeviceManager Module	151
Functions	151
NDeviceManagerGetInfo Function	151
Files	152
NDeviceManager.h	152
6.1.4 NExtractors Library	152
6.1.4.1 NFExtractor Module	153
Functions	155
NfeExtract Function	155
NfeIsTemplateSmall Function	156
NfeMakeSmallTemplate Function	156
NfeUpdateTemplate Function	157
NfeGeneralizeEx Function	157
NfeCreate Function	158
NfeCreateEx Function	158

- NfeGeneralize Function 159
- NfeIsPalm Function 160
- NfeMakeLightTemplate Function 160
- HNFExtractor 161
- Structs, Records, Enums 161
 - NfeExtractionStatus Enumeration 161
 - NfeReturnedImage Enumeration 161
 - NfeTemplateSize Enumeration 162
- Macros 162
 - NFE_MODE_AUTHENTEC_AFS2 Macro 164
 - NFE_MODE_AUTHENTEC_AES2501B Macro 164
 - NFE_MODE_AUTHENTEC_AES4000 Macro 164
 - NFE_MODE_ATMEL_FINGERCHIP Macro 164
 - NFE_MODE_BIOMETRIKA_FX2000 Macro 164
 - NFE_MODE_BIOMETRIKA_FX3000 Macro 164
 - NFE_MODE_BMF_BLP100 Macro 164
 - NFE_MODE_CROSSMATCH_VERIFIER300 Macro 165
 - NFE_MODE_DIGENT_IZZIX Macro 165
 - NFE_MODE_DIGITALPERSONA_UAREU Macro 165
 - NFE_MODE_FUJITSU_MBF200 Macro 165
 - NFE_MODE_FUTRONIC_FS80 Macro 165
 - NFE_MODE_GENERAL Macro 165
 - NFE_MODE_IDENTICATORTECHNOLOGY_DF90 Macro 165
 - NFEP_GENERALIZATION_MIN_MINUTIA_COUNT Macro 165
 - NFE_MODE_IDENTIX_DFR2090 Macro 165
 - NFEP_MIN_MINUTIA_COUNT Macro 166
 - NFE_MODE_IDENTIX_TOUCHVIEW Macro 166
 - NFE_MODE_KEYTRONIC_SECUREDESKTOP Macro 166
 - NFE_MODE_LIGHTTUNING_LTTC500 Macro 166
 - NFE_MODE_NITGEN_FINGKEY_HAMSTER Macro 166
 - NFE_MODE_PRECISEBIOMETRICS_100CS Macro 166
 - NFE_MODE_SECUGEN_HAMSTER Macro 166
 - NFE_MODE_STARTEK_FM200 Macro 166
 - NFE_MODE_TACOMA_CMOS Macro 166
 - NFE_MODE_UPEK_TOUCHCHIP Macro 167
 - NFEP_EXTRACTED_RIDGE_COUNTS Macro 167
 - NFEP_GENERALIZATION_MAXIMAL_ROTATION Macro 167
 - NFEP_GENERALIZATION_THRESHOLD Macro 167
 - NFEP_MODE Macro 167
 - NFEP_QUALITY_THRESHOLD Macro 167
 - NFEP_RETURNED_IMAGE Macro 167
 - NFEP_TEMPLATE_SIZE Macro 167
 - NFEP_USE_QUALITY Macro 168
 - NFE_MODE_ETHENTICA Macro 168

NFE_MODE_TESTECH_BIOI Macro	168
Files	168
NFExtractor.h	168
NFExtractorParams.h	169
6.1.4.2 NExtractors Module	170
Functions	171
NExtractorsGetInfo Function	171
Files	171
NExtractors.h	171
6.1.4.3 NLExtractor Module	171
Functions	173
NleCompressEx Function	173
NleDetectFace Function	174
NleDetectFaces Function	174
NleDetectFacialFeatures Function	175
NleExtract Function	175
NleExtractNext Function	176
NleExtractNextEx Function	177
NleExtractUsingDetails Function	177
NleGeneralizeEx Function	178
NleCompress Function	178
NleCreate Function	179
NleExtractStart Function	179
NleGeneralize Function	180
HNLExtractor	180
Structs, Records, Enums	181
NleRotation Structure	181
NleTemplateSize Enumeration	181
NleDetectionDetails Structure	182
NleEyes Structure	182
NleExtractionStatus Enumeration	182
NleFace Structure	183
Macros	183
NLEP_FACE_CONFIDENCE_THRESHOLD Macro	184
NLEP_MAX_ROLL_ANGLE_DEVIATION Macro	184
NLEP_TEMPLATE_SIZE Macro	185
NLEP_FACE_QUALITY_THRESHOLD Macro	185
NLEP_FAVOR_LARGEST_FACE Macro	185
NLEP_LIVENESS_THRESHOLD Macro	185
NLEP_MAX_IOD Macro	185
NLEP_MAX_RECORDS_PER_TEMPLATE Macro	186
NLEP_MIN_IOD Macro	186
NLEP_USE_LIVENESS_CHECK Macro	186
Files	186

NLExtractor.h	186
NLExtractorParams.h	187
6.1.5 NImages Library	188
6.1.5.1 Bmp Module	189
Functions	189
BmpLoadImageFromFile Function	189
BmpLoadImageFromHBitmap Function	190
BmpLoadImageFromMemory Function	190
BmpLoadImageFromStream Function	191
BmpSaveImageToFile Function	191
BmpSaveImageToHBitmap Function	192
BmpSaveImageToMemory Function	192
BmpSaveImageToStream Function	193
Files	193
Bmp.h	194
6.1.5.2 IHead Module	194
Functions	194
IHeadLoadImageFromFile Function	195
IHeadLoadImageFromMemory Function	195
IHeadLoadImageFromStream Function	196
IHeadSaveImageToFile Function	196
IHeadSaveImageToMemory Function	197
IHeadSaveImageToStream Function	197
Files	198
IHead.h	198
6.1.5.3 Jpeg Module	198
Functions	199
JpegLoadImageFromFile Function	199
JpegLoadImageFromMemory Function	199
JpegLoadImageFromStream Function	200
JpegSaveImageToFile Function	201
JpegSaveImageToMemory Function	201
JpegSaveImageToStream Function	202
LosslessJpegSaveImageToFile Function	202
LosslessJpegSaveImageToMemory Function	203
LosslessJpegSaveImageToStream Function	203
Macros	203
JPEG_DEFAULT_QUALITY Macro	204
Files	204
Jpeg.h	204
6.1.5.4 NGrayscaleImage Module	204
Files	205
NGrayscaleImage.h	205
6.1.5.5 NImage Module	205

Functions	206
NImageClone Function	206
NImageCopy Function	206
NImageCreate Function	207
NImageCreateFromData Function	208
NImageCreateFromDataPart Function	209
NImageCreateFromFile Function	210
NImageCreateFromImage Function	211
NImageCreateFromImageEx Function	212
NImageCreateWrapper Function	213
NImageCrop Function	214
NImageFlipDiagonally Function	214
NImageFlipHorizontally Function	215
NImageFlipVertically Function	215
NImageGetHeight Function	216
NImageGetHorzResolution Function	216
NImageGetPixelFormat Function	216
NImageGetPixels Function	217
NImageGetSize Function	217
NImageGetStride Function	218
NImageGetVertResolution Function	219
NImageGetWidth Function	219
NImageRotateFlip Function	219
NImageSaveToFile Function	220
HNImage	221
Structs, Records, Enums	221
NImageRotateFlipType Enumeration	221
Files	222
NImage.h	222
6.1.5.6 NImageFile Module	223
Functions	223
NImageFileClose Function	223
NImageFileCreate Function	224
NImageFileGetFormat Function	224
NImageFileIsOpened Function	225
NImageFileReadImage Function	225
HNImageFile	226
Files	226
NImageFile.h	226
6.1.5.7 NImages Module	227
Functions	227
NImagesGetGrayscaleColorWrapperEx Function	227
NImagesGetInfo Function	228
Files	228

NImages.h	228
6.1.5.8 NMonochromeImage Module	228
Functions	229
NMonochromeImageGetPixel Function	229
NMonochromeImageSetPixel Function	229
Files	230
NMonochromeImage.h	230
6.1.5.9 NPixelFormat Module	230
Functions	231
NPixelFormatIsValid Function	231
Structs, Records, Enums	232
NPixelFormat_Enumeration	232
NRGB Structure	232
Types	232
NPixelFormat Type	232
Macros	233
NCalcRowSize Macro	233
NCalcRowSizeEx Macro	233
NPixelFormatGetBitsPerPixel Macro	233
NPixelFormatGetRowSize Macro	233
NPixelFormatGetRowSizeEx Macro	233
NRGBConst Macro	234
Files	234
NPixelFormat.h	234
6.1.5.10 NRGBImage Module	235
Functions	235
NRGBImageGetPixel Function	235
NRGBImageSetPixel Function	236
Files	236
NRGBImage.h	236
6.1.5.11 Png Module	237
Functions	237
PngLoadImageFromFile Function	237
PngLoadImageFromMemory Function	238
PngLoadImageFromStream Function	238
PngSaveImageToFile Function	239
PngSaveImageToMemory Function	239
PngSaveImageToStream Function	240
Macros	241
PNG_DEFAULT_COMPRESSION_LEVEL Macro	241
Files	241
Png.h	241
6.1.5.12 Tiff Module	242
Functions	242

TiffLoadImageFromFile Function	242
TiffLoadImageFromMemory Function	243
TiffLoadImageFromStream Function	243
Files	244
Tiff.h	244
6.1.5.13 NImageFormat Module	244
Functions	245
NImageFormatCanRead Function	245
NImageFormatCanWrite Function	246
NImageFormatCanWriteMultiple Function	246
NImageFormatGetBmp Function	247
NImageFormatGetDefaultFileExtension Function	247
NImageFormatGetFileFilter Function	248
NImageFormatGetFormat Function	248
NImageFormatGetFormatCount Function	249
NImageFormatGetIHead Function	249
NImageFormatGetJpeg Function	249
NImageFormatGetJpeg2K Function	250
NImageFormatGetName Function	250
NImageFormatGetPng Function	251
NImageFormatOpenFile Function	251
NImageFormatGetTiff Function	252
NImageFormatGetWsq Function	252
NImageFormatLoadImageFromFile Function	253
NImageFormatSaveImagesToFile Function	253
NImageFormatLoadImageFromMemory Function	254
NImageFormatLoadImageFromStream Function	254
NImageFormatOpenFileFromMemory Function	255
NImageFormatOpenFileFromStream Function	256
NImageFormatSaveImageToFile Function	256
NImageFormatSaveImageToMemory Function	257
NImageFormatSaveImageToStream Function	257
NImageFormatSelect Function	258
HNImageFormat	259
Files	259
NImageFormat.h	259
6.1.6 NLicensing Library	260
6.1.6.1 NLicensing Module	261
Functions	261
NLicenseGetInfo Function	262
NLicenseIsComponentActivated Function	262
NLicenseObtain Function	263
NLicenseRelease Function	265
NLicensingGetInfo Function	266

Structs, Records, Enums	267
NLicenseInfo Structure	267
Files	267
NLicensing.h	267
6.1.7 NSmartCards Library	268
6.1.7.1 BerTlv Module	268
Functions	272
BerTagCreate Function	272
BerTagGetClass Function	273
BerTagGetEncoding Function	273
BerTagGetLength Function	273
BerTagGetNumber Function	274
BerTagIsValid Function	274
BerTagRead Function	274
BerTagWrite Function	275
BerTlvClone Function	275
BerTlvCreate Function	276
BerTlvCreateFromMemory Function	276
BerTlvGetLength Function	277
BerTlvGetMinLengthSize Function	277
BerTlvGetSize Function	278
BerTlvGetSizeMany Function	278
BerTlvGetTag Function	279
BerTlvLoadManyFromMemory Function	279
BerTlvSaveManyToMemory Function	280
BerTlvSaveToMemory Function	280
BerTlvSetMinLengthSize Function	281
ConstructedBerTlvAddDataObject Function	281
ConstructedBerTlvAddDataObjectCopy Function	282
ConstructedBerTlvAddDataObjectFromMemory Function	282
ConstructedBerTlvClearDataObjects Function	283
ConstructedBerTlvFindDataObject Function	283
ConstructedBerTlvFindDataObjectIndex Function	284
ConstructedBerTlvGetDataObject Function	285
ConstructedBerTlvGetDataObjectCapacity Function	285
ConstructedBerTlvGetDataObjectCount Function	286
ConstructedBerTlvRemoveDataObject Function	286
ConstructedBerTlvSetDataObjectCapacity Function	287
PrimitiveBerTlvGetValue Function	287
PrimitiveBerTlvGetValueAsByte Function	287
PrimitiveBerTlvGetValueAsBoolean Function	288
PrimitiveBerTlvGetValueAsInt16 Function	288
PrimitiveBerTlvGetValueAsInt32 Function	289
PrimitiveBerTlvGetValueAsInteger Function	289

PrimitiveBerTlvGetValueAsSByte Function	290
PrimitiveBerTlvGetValueAsUInt16 Function	290
PrimitiveBerTlvGetValueAsUInt32 Function	291
PrimitiveBerTlvGetValuePtr Function	291
PrimitiveBerTlvSetLength Function	292
PrimitiveBerTlvSetValue Function	292
PrimitiveBerTlvSetValueAsByte Function	293
PrimitiveBerTlvSetValueAsBoolean Function	293
PrimitiveBerTlvSetValueAsInt16 Function	294
PrimitiveBerTlvSetValueAsInt32 Function	294
PrimitiveBerTlvSetValueAsInteger Function	295
PrimitiveBerTlvSetValueAsSByte Function	295
PrimitiveBerTlvSetValueAsUInt16 Function	296
PrimitiveBerTlvSetValueAsUInt32 Function	296
HBerTlv	297
Structs, Records, Enums	297
BerTagClass Enumeration	297
BerTagEncoding Enumeration	297
Macros	297
BER_TAG_BIT_STRING Macro	297
BER_TAG_BMP_STRING Macro	298
BER_TAG_BOOLEAN Macro	298
BER_TAG_CHARACTER_STRING Macro	298
BER_TAG_EMBEDDED_PDV Macro	298
BER_TAG_END_OF_CONTENTS Macro	298
BER_TAG_ENUMERATED Macro	298
BER_TAG_EXTERNAL Macro	299
BER_TAG_GENERAL_STRING Macro	299
BER_TAG_GENERALIZED_TIME Macro	299
BER_TAG_GRAPHIC_STRING Macro	299
BER_TAG_IA5_STRING Macro	299
BER_TAG_INSTANCE_OF Macro	299
BER_TAG_INTEGER Macro	300
BER_TAG_ISO646_STRING Macro	300
BER_TAG_NULL Macro	300
BER_TAG_NUMERIC_STRING Macro	300
BER_TAG_OBJECT_DESCRIPTOR Macro	300
BER_TAG_OBJECT_IDENTIFIER Macro	301
BER_TAG_OCTET_STRING Macro	301
BER_TAG_PRINTABLE_STRING Macro	301
BER_TAG_REAL Macro	301
BER_TAG_RELATIVE_OBJECT_IDENTIFIER Macro	301
BER_TAG_SEQUENCE Macro	301
BER_TAG_SEQUENCE_OF Macro	302

BER_TAG_SET Macro	302
BER_TAG_SET_OF Macro	302
BER_TAG_T61_STRING Macro	302
BER_TAG_TELETEX_STRING Macro	302
BER_TAG_UNIVERSAL_STRING Macro	302
BER_TAG_UTC_TIME Macro	303
BER_TAG_UTF8_STRING Macro	303
BER_TAG_VIDEOTEX_STRING Macro	303
BER_TAG_VISIBLE_STRING Macro	303
Files	303
BerTlv.h	303
6.1.7.2 NSmartCardsBiometry Module	307
Structs, Records, Enums	310
SCardBiometricSubtype Enumeration	310
SCardBiometricType Enumeration	311
SCardChallengeQualifier Enumeration	312
Macros	312
SCARD_BDT_CT_TAG_CHALLENGE Macro	312
SCARD_BDT_CT_TAG_CHALLENGE_QUALIFIER Macro	312
SCARD_BDT_TAG_CHALLENGE_DATA Macro	312
SCARD_BDT_TAG_CHALLENGE_TEMPLATE Macro	313
SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA Macro	313
SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA_OBJECTS Macro	313
SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA Macro	313
SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA_OBJECTS Macro	313
SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_DATA Macro	314
SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_TEMPLATE Macro	314
SCARD_BIT_BHT_TAG_BIOMETRIC_SUBTYPE Macro	314
SCARD_BIT_BHT_TAG_BIOMETRIC_TYPE Macro	314
SCARD_BIT_BHT_TAG_CREATION_DATE Macro	314
SCARD_BIT_BHT_TAG_CREATOR Macro	315
SCARD_BIT_BHT_TAG_FORMAT_OWNER Macro	315
SCARD_BIT_BHT_TAG_FORMAT_TYPE Macro	315
SCARD_BIT_BHT_TAG_INDEX Macro	315
SCARD_BIT_BHT_TAG_PATRON_HEADER_VERSION Macro	315
SCARD_BIT_BHT_TAG_PRODUCT_IDENTIFIER Macro	316
SCARD_BIT_BHT_TAG_PROPRIETARY_BHT Macro	316
SCARD_BIT_BHT_TAG_STANDARD_BHT Macro	316
SCARD_BIT_BHT_TAG_VALIDITY_PERIOD Macro	316
SCARD_BIT_TAG_ALGORITHM_REFERENCE Macro	316
SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE Macro	317
SCARD_BIT_TAG_REFERENCE_DATA_QUALIFIER Macro	317
SCARD_BIT_TAG_STANDARD_DATA_OBJECTS Macro	317
SCARD_BIT_TAG_TAG_ALLOCATION_AUTHORITY_DATA_OBJECTS Macro	317

Files	317
NSmartCardsBiometry.h	318
6.1.7.3 NSmartCardsCommands Module	320
Functions	371
ApuClassCreate Function	371
ApuClassGetChannelNumber Function	372
ApuClassGetSecureMessaging Function	372
ApuClassIsInterindustry Function	372
ApuClassIsInterindustryFirst Function	373
ApuClassIsInterindustryFurther Function	373
ApuClassIsInterindustryReserved Function	373
ApuClassIsLastOrTheOnly Function	374
ApuClassIsProprietary Function	374
ApuClassIsValid Function	374
ApuInstructionIsDataBerTlvEncoded Function	375
ApuInstructionIsValid Function	375
ApuStatusGetDataStillAvailableValue Function	375
ApuStatusGetErrorSonvmuTriggeringByTheCardValue Function	376
ApuStatusGetErrorWrongLeValue Function	376
ApuStatusGetWarningSonvmcCounterValue Function	376
ApuStatusGetWarningSonvmuTriggeringByTheCardValue Function	377
ApuStatusIsCheckingError Function	377
ApuStatusIsExecutionError Function	377
ApuStatusIsInterindustry Function	378
ApuStatusIsNormalProcessing Function	378
ApuStatusIsProcessAborted Function	378
ApuStatusIsProcessComplete Function	378
ApuStatusIsProprietary Function	379
ApuStatusIsStateOfNonVolatileMemoryChanged Function	379
ApuStatusIsStateOfNonVolatileMemoryUnchanged Function	379
ApuStatusIsValid Function	380
ApuStatusIsWarningProcessing Function	380
SCardTransmitApu Function	380
Structs, Records, Enums	381
ApuClassSecureMessaging Enumeration	381
Macros	382
APDU_INS_ACTIVATE_FILE Macro	382
APDU_INS_APPEND_RECORD Macro	382
APDU_INS_CHANGE_REFERENCE_DATA Macro	382
APDU_INS_CREATE_FILE Macro	382
APDU_INS_DEACTIVATE_FILE Macro	383
APDU_INS_DELETE_FILE Macro	383
APDU_INS_DISABLE_VERIFICATION_REQUIREMENT Macro	383
APDU_INS_ENABLE_VERIFICATION_REQUIREMENT Macro	383

APDU_INS_ENVELOPE Macro 383
APDU_INS_ENVELOPE_BER_TLV Macro 383
APDU_INS_ERASE_BINARY Macro 384
APDU_INS_ERASE_BINARY_BER_TLV Macro 384
APDU_INS_ERASE_RECORD Macro 384
APDU_INS_EXTERNAL_AUTHENTICATE Macro 384
APDU_INS_GENERAL_AUTHENTICATE Macro 384
APDU_INS_GENERAL_AUTHENTICATE_BER_TLV Macro 384
APDU_INS_GENERATE_ASYMMETRIC_KEY_PAIR Macro 385
APDU_INS_GET_CHALLENGE Macro 385
APDU_INS_GET_DATA Macro 385
APDU_INS_GET_DATA_BER_TLV Macro 385
APDU_INS_GET_RESPONSE Macro 385
APDU_INS_INTERNAL_AUTHENTICATE Macro 386
APDU_INS_MANAGE_CHANNEL Macro 386
APDU_INS_MANAGE_SECURITY_ENVIRONMENT Macro 386
APDU_INS_PERFORM_SCQL_OPERATION Macro 386
APDU_INS_PERFORM_SECURITY_OPERATION Macro 386
APDU_INS_PERFORM_TRANSACTION_OPERATION Macro 386
APDU_INS_PERFORM_USER_OPERATION Macro 387
APDU_INS_PUT_DATA Macro 387
APDU_INS_PUT_DATA_BER_TLV Macro 387
APDU_INS_READ_BINARY Macro 387
APDU_INS_READ_BINARY_BER_TLV Macro 387
APDU_INS_READ_RECORD Macro 387
APDU_INS_READ_RECORD_BER_TLV Macro 388
APDU_INS_RESET_RETRY_COUNTER Macro 388
APDU_INS_SEARCH_BINARY Macro 388
APDU_INS_SEARCH_BINARY_BER_TLV Macro 388
APDU_INS_SEARCH_RECORD Macro 388
APDU_INS_SELECT Macro 389
APDU_INS_TERMINATE_CARD_USAGE Macro 389
APDU_INS_TERMINATE_DF Macro 389
APDU_INS_TERMINATE_EF Macro 389
APDU_INS_UPDATE_BINARY Macro 389
APDU_INS_UPDATE_BINARY_BER_TLV Macro 389
APDU_INS_UPDATE_RECORD Macro 390
APDU_INS_UPDATE_RECORD_BER_TLV Macro 390
APDU_INS_VERIFY Macro 390
APDU_INS_VERIFY_BER_TLV Macro 390
APDU_INS_WRITE_BINARY Macro 390
APDU_INS_WRITE_BINARY_BER_TLV Macro 390
APDU_INS_WRITE_RECORD Macro 391
APDU_STATUS_DATA_STILL_AVAILABLE_XX Macro 391

APDU_STATUS_ERROR_CLASS_NOT_SUPPORTED Macro 391

APDU_STATUS_ERROR_CNA_AUTHENTICATION_METHOD_BLOCKED Macro 391

APDU_STATUS_ERROR_CNA_COMMAND_INCOMPATIBLE_WITH_FILE_STRUCTURE Macro 391

APDU_STATUS_ERROR_CNA_COMMAND_NOT_ALLOWED Macro 392

APDU_STATUS_ERROR_CNA_CONDITIONS_OF_USE_NOT_SATISFIED Macro 392

APDU_STATUS_ERROR_CNA_EXPECTED_SECURE_MESSAGING_DATA_OBJECTS_MISSING Macro 392

APDU_STATUS_ERROR_CNA_INCORRECT_SECURE_MESSAGING_DATA_OBJECTS Macro 392

APDU_STATUS_ERROR_CNA_NO_INFORMATION Macro 393

APDU_STATUS_ERROR_CNA_REFERENCE_DATA_NOT_USABLE Macro 393

APDU_STATUS_ERROR_CNA_SECURITY_STATUS_NOT_SATISFIED Macro 393

APDU_STATUS_ERROR_CNA_XX Macro 393

APDU_STATUS_ERROR_FICNS_COMMAND_CHAINING_NOT_SUPPORTED Macro 393

APDU_STATUS_ERROR_FICNS_LAST_COMMAND_OF_THE_CHAIN_EXPECTED Macro 394

APDU_STATUS_ERROR_FICNS_LOGICAL_CHANNEL_NOT_SUPPORTED Macro 394

APDU_STATUS_ERROR_FICNS_NO_INFORMATION Macro 394

APDU_STATUS_ERROR_FICNS_SECURE_MESSAGING_NOT_SUPPORTED Macro 394

APDU_STATUS_ERROR_FICNS_XX Macro 394

APDU_STATUS_ERROR_INSTRUCTION_NOT_SUPPORTED_OR_INVALID Macro 395

APDU_STATUS_ERROR_NO_PRECISE_DIAGNOSIS Macro 395

APDU_STATUS_ERROR_PROPRIETARY_67XX Macro 395

APDU_STATUS_ERROR_PROPRIETARY_6BXX Macro 395

APDU_STATUS_ERROR_PROPRIETARY_6DXX Macro 395

APDU_STATUS_ERROR_PROPRIETARY_6EXX Macro 396

APDU_STATUS_ERROR_PROPRIETARY_6FXX Macro 396

APDU_STATUS_ERROR_SECURITY_XX Macro 396

APDU_STATUS_ERROR_SONVMC_MEMORY_FAILURE Macro 396

APDU_STATUS_ERROR_SONVMC_NO_INFORMATION Macro 396

APDU_STATUS_ERROR_SONVMC_XX Macro 396

APDU_STATUS_ERROR_SONVMU_EXECUTION_ERROR Macro 397

APDU_STATUS_ERROR_SONVMU_IMMEDIATE_RESPONSE_REQUIRED_BY_THE_CARD Macro 397

APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_END Macro 397

APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_START Macro 397

APDU_STATUS_ERROR_SONVMU_XX Macro 398

APDU_STATUS_ERROR_WP1P2_DF_NAME_ALREADY_EXISTS Macro 398

APDU_STATUS_ERROR_WP1P2_FILE_ALREADY_EXISTS Macro 398

APDU_STATUS_ERROR_WP1P2_FILE_OR_APPLICATION_NOT_FOUND Macro 398

APDU_STATUS_ERROR_WP1P2_FUNCTION_NOT_SUPPORTED Macro 398

APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_IN_THE_COMMAND_DATA_FIELD Macro 399

APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_P1P2 Macro 399

APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_PARAMETERS_P1P2 Macro 399

APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_TLV_STRUCTURE Macro 399

APDU_STATUS_ERROR_WP1P2_NO_INFORMATION Macro 399

APDU_STATUS_ERROR_WP1P2_NO_QUALIFICATION Macro 400

APDU_STATUS_ERROR_WP1P2_NOT_ENOUGH_MEMORY_SPACE_IN_THE_FILE Macro 400

APDU_STATUS_ERROR_WP1P2_RECORD_NOT_FOUND Macro 400
APDU_STATUS_ERROR_WP1P2_REFERENCED_DATA_OR_REFERENCE_DATA_NOT_FOUND Macro 400
APDU_STATUS_ERROR_WP1P2_XX Macro 400
APDU_STATUS_ERROR_WRONG_LE_XX Macro 401
APDU_STATUS_ERROR_WRONG_LENGTH Macro 401
APDU_STATUS_NO_QUALIFICATION Macro 401
APDU_STATUS_PROPRIETARY_9XXX Macro 401
APDU_STATUS_WARNING_SONVMC_COUNTER_X Macro 401
APDU_STATUS_WARNING_SONVMC_FILE_FILLED_UP_BY_THE_LAST_WRITE Macro 402
APDU_STATUS_WARNING_SONVMC_NO_INFORMATION Macro 402
APDU_STATUS_WARNING_SONVMC_XX Macro 402
APDU_STATUS_WARNING_SONVMU_END_OF_FILE_OR_RECORD_BEFORE_READING_NE_BYTES Macro 402
APDU_STATUS_WARNING_SONVMU_FILE_CONTROL_INFORMATION_INCORRECTY_FORMATTED Macro 402
APDU_STATUS_WARNING_SONVMU_NO_INFORMATION Macro 403
APDU_STATUS_WARNING_SONVMU_NO_INPUT_DATA_AVAILABLE_FROM_A_SENSOR_ON_THE_CARD Macro 403
APDU_STATUS_WARNING_SONVMU_PART_OF_RETURNED_DATA_MAY_BE_CORRUPTED Macro 403
APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_DEACTIVATED Macro 403
APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_IN_TERMINATION_STATE Macro 404
APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_END Macro 404
APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_START Macro 404
APDU_STATUS_WARNING_SONVMU_XX Macro 404
ApuStatusGetSW1 Macro 404
ApuStatusGetSW2 Macro 405
ApuStatusIsDataStillAvailable Macro 405
ApuStatusIsErrorCna Macro 405
ApuStatusIsErrorFicns Macro 405
ApuStatusIsErrorProprietary67XX Macro 405
ApuStatusIsErrorProprietary6BXX Macro 406
ApuStatusIsErrorProprietary6DXX Macro 406
ApuStatusIsErrorProprietary6EXX Macro 406
ApuStatusIsErrorProprietary6FXX Macro 406
ApuStatusIsErrorSecurity Macro 406
ApuStatusIsErrorSonvmc Macro 407
ApuStatusIsErrorSonvmu Macro 407
ApuStatusIsErrorSonvmuTriggeringByTheCard Macro 407
ApuStatusIsErrorWP1P2 Macro 407
ApuStatusIsErrorWrongLe Macro 407
ApuStatusIsProprietary9XXX Macro 408
ApuStatusIsWarningSonvmc Macro 408
ApuStatusIsWarningSonvmcCounter Macro 408
ApuStatusIsWarningSonvmu Macro 408
ApuStatusIsWarningSonvmuTriggeringByTheCard Macro 408

Files 409

NSmartCardsCommands.h 409

6.1.7.4 NSmartCardsDataElements Module 459

Macros 471

SCARD_TAG_ACCOUNT_TYPE Macro 471
SCARD_TAG_ADDRESS Macro 471
SCARD_TAG_ANSWER_TO_RESET Macro 471
SCARD_TAG_APPLICATION_EFFECTIVE_DATE Macro 471
SCARD_TAG_APPLICATION_EXPIRATION_DATE Macro 472
SCARD_TAG_APPLICATION_FAMILY_IDENTIFIER Macro 472
SCARD_TAG_APPLICATION_IDENTIFIER Macro 472
SCARD_TAG_APPLICATION_IMAGE Macro 472
SCARD_TAG_APPLICATION_IMAGE_TEMPLATE Macro 472
SCARD_TAG_APPLICATION_LABEL Macro 473
SCARD_TAG_APPLICATION_RELATED_DATA Macro 473
SCARD_TAG_APPLICATION_TEMPLATE Macro 473
SCARD_TAG_APPLICATION_TRACK_1 Macro 473
SCARD_TAG_APPLICATION_TRACK_2 Macro 473
SCARD_TAG_APPLICATION_TRACK_3 Macro 473
SCARD_TAG_AUTHENTICATION_DATA Macro 474
SCARD_TAG_BANK_IDENTIFIER_CODE Macro 474
SCARD_TAG_BIOMETRIC_DATA_TEMPLATE Macro 474
SCARD_TAG_BIOMETRIC_INFORMATION_GROUP_TEMPLATE Macro 474
SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE Macro 474
SCARD_TAG_CARD_CAPABILITIES Macro 475
SCARD_TAG_CARD_DATA Macro 475
SCARD_TAG_CARD_EFFECTIVE_DATE Macro 475
SCARD_TAG_CARD_EXPIRATION_DATE Macro 475
SCARD_TAG_CARD_ISSUERS_DATA Macro 475
SCARD_TAG_CARD_SEQUENCE_NUMBER Macro 475
SCARD_TAG_CARD_SERVICE_DATA Macro 476
SCARD_TAG_CARD_TRACK_1 Macro 476
SCARD_TAG_CARD_TRACK_2 Macro 476
SCARD_TAG_CARD_TRACK_3 Macro 476
SCARD_TAG_CARDHOLDER_BIOMETRIC_DATA Macro 476
SCARD_TAG_CARDHOLDER_CERTIFICATE Macro 477
SCARD_TAG_CARDHOLDER_HANDWRITTEN_SIGNATURE_IMAGE Macro 477
SCARD_TAG_CARDHOLDER_IMAGE_TEMPLATE Macro 477
SCARD_TAG_CARDHOLDER_NAME Macro 477
SCARD_TAG_CARDHOLDER_NATIONALITY Macro 477
SCARD_TAG_CARDHOLDER_PORTRAIT_IMAGE Macro 478
SCARD_TAG_CARDHOLDER_PRIVATE_KEY Macro 478
SCARD_TAG_CARDHOLDER_PRIVATE_KEY_TEMPLATE Macro 478
SCARD_TAG_CARDHOLDER_PUBLIC_KEY Macro 478
SCARD_TAG_CARDHOLDER_PUBLIC_KEY_TEMPLATE Macro 478

SCARD_TAG_CARDHOLDER_RELATED_DATA Macro 478
SCARD_TAG_CARDHOLDER_REQUIREMENTS_EXCLUDED_FEATURES Macro 479
SCARD_TAG_CARDHOLDER_REQUIREMENTS_INCLUDED_FEATURES Macro 479
SCARD_TAG_CERTIFICATE_CONTENT Macro 479
SCARD_TAG_CERTIFICATE_CONTENT_TEMPLATE Macro 479
SCARD_TAG_CERTIFICATE HOLDER_AUTHORIZATION Macro 479
SCARD_TAG_COEXISTENT_TAG_ALLOCATION_AUTHORITY Macro 480
SCARD_TAG_COMMAND_TO_PERFORM Macro 480
SCARD_TAG_COMPATIBLE_TAG_ALLOCATION_AUTHORITY Macro 480
SCARD_TAG_COUNTRY_CODE Macro 480
SCARD_TAG_COUNTRY_CODE_ALPHA2_FORMAT Macro 480
SCARD_TAG_COUNTRY_CODE_ALPHA3_FORMAT Macro 481
SCARD_TAG_COUNTRY_CODE_AND_NATIONAL_DATA Macro 481
SCARD_TAG_CURRENCY_CODE Macro 481
SCARD_TAG_CURRENCY_EXPONENT Macro 481
SCARD_TAG_DATE_OF_BIRTH Macro 481
SCARD_TAG_DEPRECATED Macro 482
SCARD_TAG_DIGITAL_SIGNATURE Macro 482
SCARD_TAG_DIGITAL_SIGNATURE_BLOCK Macro 482
SCARD_TAG_DISCRETIONARY_DATA Macro 482
SCARD_TAG_DISCRETIONARY_DATA_OBJECTS Macro 482
SCARD_TAG_DISPLAY_CONTROL Macro 482
SCARD_TAG_DISPLAY_MESSAGE Macro 483
SCARD_TAG_DYNAMIC_AUTHENTICATION_TEMPLATE Macro 483
SCARD_TAG_DYNAMIC_EXTERNAL_AUTHENTICATION Macro 483
SCARD_TAG_DYNAMIC_INTERNAL_AUTHENTICATION Macro 483
SCARD_TAG_DYNAMIC_MUTUAL_AUTHENTICATION Macro 483
SCARD_TAG_ELEMENT_LIST Macro 484
SCARD_TAG_EXTENDED_HEADER_LIST Macro 484
SCARD_TAG_FCI_TEMPLATE Macro 484
SCARD_TAG_FCP_TEMPLATE Macro 484
SCARD_TAG_FILE_REFERENCE Macro 484
SCARD_TAG_FMD_TEMPLATE Macro 484
SCARD_TAG_HEADER_LIST Macro 485
SCARD_TAG_HISTORICAL_BYTES Macro 485
SCARD_TAG_INITIAL_ACCESS_DATA Macro 485
SCARD_TAG_INTEGRATED_CIRCUIT_MANUFACTURER_IDENTIFIER Macro 485
SCARD_TAG_INTERCHANGE_CONTROL Macro 485
SCARD_TAG_INTERCHANGE_PROFILE Macro 486
SCARD_TAG_INTERINDUSTRY_DATA_OBJECTS Macro 486
SCARD_TAG_INTERNATIONAL_BANK_ACCOUNT_NUMBER Macro 486
SCARD_TAG_ISSUER_IDENTIFICATION_NUMBER Macro 486
SCARD_TAG_LANGUAGE_PREFERENCES Macro 486
SCARD_TAG_LOGIN_DATA_PROPRIETARY Macro 486

SCARD_TAG_LOGIN_TEMPLATE Macro	487
SCARD_TAG_MESSAGE_REFERENCE Macro	487
SCARD_TAG_NAME Macro	487
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_70 Macro	487
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_71 Macro	487
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_72 Macro	488
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_74 Macro	488
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_75 Macro	488
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_76 Macro	488
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_77 Macro	488
SCARD_TAG_OFFSET_DATA_OBJECT Macro	488
SCARD_TAG_PIN_USAGE_POLICY Macro	489
SCARD_TAG_PRE_ISSUING_DATA Macro	489
SCARD_TAG_PRIMARY_ACCOUNT_NUMBER Macro	489
SCARD_TAG_PUBLIC_KEY_OF_CERTIFICATION_AUTHORITY Macro	489
SCARD_TAG_QUALIFIED_NAME Macro	489
SCARD_TAG_SECURE_MESSAGING_TEMPLATE Macro	490
SCARD_TAG_SECURITY_ENVIRONMENT_TEMPLATE Macro	490
SCARD_TAG_SECURITY_SUPPORT_TEMPLATE Macro	490
SCARD_TAG_SERVICE_CODE Macro	490
SCARD_TAG_SEX Macro	490
SCARD_TAG_SPECIAL_USER_REQUIREMENTS Macro	490
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION Macro	491
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA Macro	491
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA Macro	491
SCARD_TAG_STATUS_INFORMATION Macro	491
SCARD_TAG_TAG_LIST Macro	492
SCARD_TAG_TIMER Macro	492
SCARD_TAG_TRANSACTION_COUNTER Macro	492
SCARD_TAG_TRANSACTION_DATE Macro	492
SCARD_TAG_UNIFORM_RESOURCE_LOCATOR Macro	492
SCARD_TAG_WRAPPER Macro	493
Files	493
NSmartCardsDataElements.h	493
6.1.7.5 SimpleTlv Module	504
Functions	505
SimpleTagIsValid Function	505
SimpleTlvClone Function	505
SimpleTlvCreate Function	506
SimpleTlvCreateFromMemory Function	506
SimpleTlvGetLength Function	507
SimpleTlvGetSize Function	507
SimpleTlvGetSizeMany Function	508
SimpleTlvGetTag Function	508

SimpleTlvGetUseThreeByteLengthAlways Function	509
SimpleTlvGetValue Function	509
SimpleTlvGetValuePtr Function	510
SimpleTlvLoadManyFromMemory Function	510
SimpleTlvSaveManyToMemory Function	511
SimpleTlvSaveToMemory Function	511
SimpleTlvSetLength Function	512
SimpleTlvSetUseThreeByteLengthAlways Function	512
SimpleTlvSetValue Function	513
HSimpleTlv	513
Files	513
SimpleTlv.h	513
6.1.8 NTemplates Library	514
6.1.8.1 NFRecordV1 Module	515
Functions	515
NFRecordGetMaxSizeV1 Function	515
NFRecordGetSizeV1 Function	516
NFRecordSaveToMemoryV1 Function	517
Files	518
NFRecordV1.h	518
6.1.8.2 NFRecord Module	518
Functions	523
NFRecordAddCore Function	523
NFRecordGetMaxSize Function	524
NFRecordAddDelta Function	525
NFRecordAddDoubleCore Function	526
NFRecordAddMinutia Function	526
NFRecordCheck Function	527
NFRecordClearCores Function	527
NFRecordClearDeltas Function	527
NFRecordClearDoubleCores Function	528
NFRecordClearMinutiae Function	528
NFRecordClone Function	529
NFRecordCreate Function	529
NFRecordCreateEx Function	530
NFRecordCreateFromMemory Function	531
NFRecordGetCbeffProductType Function	532
NFRecordGetCbeffProductTypeMem Function	532
NFRecordGetCore Function	533
NFRecordGetCoreCapacity Function	533
NFRecordGetCoreCount Function	534
NFRecordGetCores Function	535
NFRecordGetDelta Function	535
NFRecordGetDeltaCapacity Function	536

NFRecordGetDeltaCount Function 536
NFRecordGetDeltas Function 537
NFRecordGetDoubleCore Function 537
NFRecordGetDoubleCoreCapacity Function 538
NFRecordGetDoubleCoreCount Function 538
NFRecordGetDoubleCores Function 539
NFRecordGetG Function 540
NFRecordGetGMem Function 540
NFRecordGetHeight Function 541
NFRecordGetHeightMem Function 541
NFRecordGetHorzResolution Function 542
NFRecordGetHorzResolutionMem Function 542
NFRecordGetImpressionType Function 543
NFRecordGetImpressionTypeMem Function 543
NFRecordGetMinutia Function 544
NFRecordGetMinutiaCapacity Function 544
NFRecordGetMinutiaCount Function 545
NFRecordGetMinutiae Function 546
NFRecordGetMinutiaFormat Function 546
NFRecordGetMinutiaNeighbor Function 547
NFRecordGetMinutiaNeighborCount Function 547
NFRecordGetMinutiaNeighbors Function 548
NFRecordGetPatternClass Function 548
NFRecordGetPatternClassMem Function 549
NFRecordGetPosition Function 550
NFRecordGetPositionMem Function 550
NFRecordGetQuality Function 551
NFRecordGetQualityMem Function 551
NFRecordGetRidgeCountsType Function 552
NFRecordGetSize Function 552
NFRecordGetVertResolution Function 553
NFRecordGetVertResolutionMem Function 553
NFRecordGetWidth Function 554
NFRecordGetWidthMem Function 554
NFRecordInfoDispose Function 555
NFRecordInsertCore Function 555
NFRecordInsertDelta Function 556
NFRecordInsertDoubleCore Function 556
NFRecordInsertMinutia Function 557
NFRecordRemoveCore Function 558
NFRecordRemoveDelta Function 558
NFRecordRemoveDoubleCore Function 559
NFRecordRemoveMinutia Function 559
NFRecordSaveToMemory Function 560

- NFRecordSetCbeffProductType Function 560
- NFRecordSetCore Function 561
- NFRecordSetCoreCapacity Function 561
- NFRecordSetDelta Function 562
- NFRecordSetDeltaCapacity Function 563
- NFRecordSetDoubleCore Function 563
- NFRecordSetDoubleCoreCapacity Function 564
- NFRecordSetG Function 564
- NFRecordSetImpressionType Function 565
- NFRecordSetMinutia Function 565
- NFRecordSetMinutiaCapacity Function 566
- NFRecordSetMinutiaFormat Function 566
- NFRecordSetMinutiaNeighbor Function 567
- NFRecordSetPatternClass Function 567
- NFRecordSetPosition Function 568
- NFRecordSetQuality Function 568
- NFRecordSetRidgeCountsType Function 569
- NFRecordSortMinutiae Function 569
- NFRecordTruncateMinutiae Function 570
- NFRecordTruncateMinutiaeByQuality Function 570
- HNFRecord 571
- Structs, Records, Enums 571
 - NFCore Structure 571
 - NFDelta Structure 571
 - NFDoubleCore Structure 572
 - NFImpressionType Enumeration 572
 - NFMinutia Structure 573
 - NFMinutiaFormat Enumeration 574
 - NFMinutiaNeighbor Structure 574
 - NFMinutiaOrder Enumeration 574
 - NFMinutiaType Enumeration 575
 - NFPatternClass Enumeration 575
 - NFPosition Enumeration 576
 - NFRecordInfo Structure 578
 - NFRidgeCountsType Enumeration 578
- Macros 578
 - NFR_MAX_FINGER_CORE_COUNT Macro 579
 - NFR_MAX_FINGER_DELTA_COUNT Macro 579
 - NFR_MAX_DIMENSION Macro 579
 - NFR_MAX_DOUBLE_CORE_COUNT Macro 579
 - NFR_MAX_MINUTIA_COUNT Macro 579
 - NFR_RESOLUTION Macro 579
 - NFR_SAVE_BLOCKED_ORIENTS Macro 580
 - NFR_SKIP_BLOCKED_ORIENTS Macro 580

NFR_SKIP_CURVATURES Macro	580
NFR_SKIP_GS Macro	580
NFR_SKIP_QUALITIES Macro	580
NFR_SKIP RIDGE_COUNTS Macro	580
NFR_SKIP_SINGULAR_POINTS Macro	581
NFR_SAVE_V2 Macro	581
NFR_SAVE_V3 Macro	581
NFR_MAX_CORE_COUNT Macro	581
NFR_MAX_DELTA_COUNT Macro	581
NFR_MAX_FINGER_DIMENSION Macro	582
NFR_MAX_FINGER_DOUBLE_CORE_COUNT Macro	582
NFR_MAX_FINGER_MINUTIA_COUNT Macro	582
NFR_MAX_PALM_CORE_COUNT Macro	582
NFR_MAX_PALM_DELTA_COUNT Macro	582
NFR_MAX_PALM_DIMENSION Macro	582
NFR_MAX_PALM_DOUBLE_CORE_COUNT Macro	583
NFR_MAX_PALM_MINUTIA_COUNT Macro	583
Files	583
NFRecord.h	583
6.1.8.3 NFTemplate Module	588
Functions	589
NFTemplateAddRecord Function	589
NFTemplateAddRecordCopy Function	590
NFTemplateAddRecordFromMemory Function	591
NFTemplateCalculateSize Function	592
NFTemplateCheck Function	592
NFTemplateClearRecords Function	593
NFTemplateClone Function	593
NFTemplateCreate Function	594
NFTemplateCreateEx Function	594
NFTemplateCreateFromMemory Function	595
NFTemplateGetRecord Function	596
NFTemplateGetRecordCapacity Function	596
NFTemplateGetRecordCount Function	597
NFTemplateGetRecordCountMem Function	597
NFTemplateGetSize Function	598
NFTemplateInfoDispose Function	598
NFTemplateIsPalm Function	599
NFTemplateIsPalmMem Function	599
NFTemplatePack Function	600
NFTemplateRemoveRecord Function	601
NFTemplateSaveToMemory Function	601
NFTemplateSetRecordCapacity Function	602
NFTemplateUnpack Function	603

HNFTemplate	604
Macros	604
NFT_MAX_RECORD_COUNT Macro	604
NFT_PROCESS_FIRST_RECORD_ONLY Macro	604
Files	604
NFTemplate.h	604
6.1.8.4 NTemplates Module	605
Functions	606
NTemplatesGetInfo Function	606
Files	606
NTemplates.h	606
6.2 .NET Reference	606
6.2.1 Neurotec Namespace	607
6.2.1.1 Classes	608
NCore Class	608
NCore Fields	609
NCore.DllName Field	609
NCore Methods	609
Alloc Method	609
CAlloc Method	609
Clear Method	610
Compare Method	610
Copy Method	611
Fill Method	612
NCore.Free Method	612
NCore.GetInfo Method	612
Move Method	613
PtrToArray Method	613
NCore.PtrToStructureArray Method	614
ReAlloc Method	614
WriteBufferToStream Method	615
NDisposable Class	615
NDisposable Methods	615
NDisposable.Dispose Method	616
NeurotecException Class	616
NeurotecExceptionBase Class	616
NeurotecExceptionBase Properties	617
NeurotecExceptionBase.Code Property	617
NeurotecExceptionBase.ManagedStackTrace Property	617
NeurotecExceptionBase.StackTrace Property	617
NeurotecExceptionBase.UnmanagedStackTrace Property	617
NIndexPair Structure	617
NIndexPair.NIndexPair Constructor	618
NIndexPair Methods	618

- NIndexPair.ToString Method 618
- NIndexPair Properties 618
 - NIndexPair.Index1 Property 618
 - NIndexPair.Index2 Property 618
- NInfo Class 619
- NInfoObject Class 619
- NLibraryInfo Class 619
 - NLibraryInfo Methods 619
 - NLibraryInfo.Retrieve Method 620
 - NLibraryInfo Properties 620
 - NLibraryInfo.Activated Property 620
 - NLibraryInfo.Company Property 620
 - NLibraryInfo.Copyright Property 620
 - NLibraryInfo.Product Property 620
 - NLibraryInfo.Title Property 620
 - NLibraryInfo.Version Property 620
- NObject Class 621
 - NObject Methods 621
 - NObject.CopyParameters Method 621
 - NObject.Free Method 621
 - NObject.GetNativeType Method 622
 - GetParameter Method 622
 - NObject.Reset Method 622
 - SetParameter Method 622
 - NObject Properties 623
 - NObject.Handle Property 623
 - NObject.Owner Property 623
- NotActivatedException Class 623
- NProcessorInfo Class 624
 - NProcessorInfo Methods 624
 - NProcessorInfo.GetModelInfo Method 625
 - NProcessorInfo Properties 625
 - NProcessorInfo.Is3DNowSupported Property 625
 - NProcessorInfo.IsLZCntSupported Property 625
 - NProcessorInfo.IsMmxSupported Property 625
 - NProcessorInfo.IsPopCntSupported Property 625
 - NProcessorInfo.IsSse2Supported Property 625
 - NProcessorInfo.IsSse3Supported Property 625
 - NProcessorInfo.IsSse41Supported Property 625
 - NProcessorInfo.IsSse42Supported Property 626
 - NProcessorInfo.IsSse4aSupported Property 626
 - NProcessorInfo.IsSse5Supported Property 626
 - NProcessorInfo.IsSseSupported Property 626
 - NProcessorInfo.IsSsse3Supported Property 626

- NProcessorInfo.ModelName Property 626
- NProcessorInfo.Vendor Property 626
- NProcessorInfo.VendorName Property 626
- NRange Structure 626
 - NRange.NRange Constructor 627
 - NRange Methods 627
 - NRange.ToString Method 627
 - NRange Properties 627
 - NRange.From Property 627
 - NRange.To Property 627
- NRational Structure 628
 - NRational.NRational Constructor 628
 - NRational Fields 628
 - NRational.Empty Field 628
 - NRational Methods 628
 - NRational.ToString Method 629
 - NRational Properties 629
 - NRational.Denominator Property 629
 - NRational.Numerator Property 629
- NResult Class 629
 - NResult Fields 630
 - NResult.EArgument Field 630
 - NResult.EArgumentNull Field 630
 - NResult.EArgumentOutOfRange Field 630
 - NResult.EArithmetic Field 631
 - NResult.EClr Field 631
 - NResult.ECom Field 631
 - NResult.ECore Field 631
 - NResult.EDirectoryNotFound Field 631
 - NResult.EDriveNotFound Field 631
 - NResult.EEndOfStream Field 631
 - NResult.EExternal Field 631
 - NResult.EFailed Field 631
 - NResult.EFileLoad Field 632
 - NResult.EFileNotFound Field 632
 - NResult.EFormat Field 632
 - NResult.EIndexOutOfRange Field 632
 - NResult.EInvalidCast Field 632
 - NResult.EInvalidEnumArgument Field 632
 - NResult.EInvalidOperation Field 632
 - NResult.EIO Field 632
 - NResult.ENotActivated Field 632
 - NResult.ENotImplemented Field 633
 - NResult.ENotSupported Field 633

NResult.ENullReference Field	633
NResult.EOutOfMemory Field	633
NResult.EOverflow Field	633
NResult.EParameter Field	633
NResult.EParameterReadOnly Field	633
NResult.EPathTooLong Field	633
NResult.ESecurity Field	633
NResult.ESys Field	634
NResult.EWin32 Field	634
NResult.Ok Field	634
NResult Methods	634
NResult.Check Method	634
NResult.IsFailed Method	634
NResult.IsSucceeded Method	634
NResult.RaiseError Method	635
NResult.SetError Method	635
NURational Structure	635
NURational.NURational Constructor	635
NURational Fields	636
NURational.Empty Field	636
NURational Methods	636
NURational.ToString Method	636
NURational Properties	636
NURational.Denominator Property	636
NURational.Numerator Property	636
ParameterException Class	636
ParameterReadOnlyException Class	637
6.2.1.2 Interfaces	637
INeurotecException Interface	637
INeurotecException Properties	638
INeurotecException.Code Property	638
INeurotecException.ManagedStackTrace Property	638
INeurotecException.UnmanagedStackTrace Property	638
6.2.1.3 Structs, Records, Enums	638
Neurotec.NProcessorVendor Enumeration	638
6.2.1.4 Types	639
Neurotec.NLibraryGetInfo Type	639
6.2.2 Neurotec.Biometrics Namespace	639
6.2.2.1 Classes	640
NExtractors Class	640
NExtractors Fields	641
NExtractors.DllName Field	641
NExtractors Methods	641
NExtractors.GetInfo Method	641

- NFCore Structure 641
 - NFCore Constructor 641
 - NFCore.NFCore Constructor (ushort, ushort) 641
 - NFCore.NFCore Constructor (ushort, ushort, double) 642
 - NFCore.NFCore Constructor (ushort, ushort, int) 642
 - NFCore Methods 642
 - NFCore.ToString Method 642
 - NFCore Properties 642
 - NFCore.Angle Property 642
 - NFCore.Y Property 643
 - NFCore.RawAngle Property 643
 - NFCore.X Property 643
- NFDelta Structure 643
 - NFDelta Constructor 644
 - NFDelta.NFDelta Constructor (ushort, ushort) 644
 - NFDelta.NFDelta Constructor (ushort, ushort, double, double, double) 644
 - NFDelta.NFDelta Constructor (ushort, ushort, int, int, int) 644
 - NFDelta Methods 645
 - NFDelta.ToString Method 645
 - NFDelta Properties 645
 - NFDelta.Angle1 Property 645
 - NFDelta.Angle2 Property 645
 - NFDelta.Angle3 Property 646
 - NFDelta.Y Property 646
 - NFDelta.RawAngle1 Property 646
 - NFDelta.RawAngle2 Property 646
 - NFDelta.RawAngle3 Property 646
 - NFDelta.X Property 647
- NFDoubleCore Structure 647
 - NFDoubleCore.NFDoubleCore Constructor 647
 - NFDoubleCore Methods 647
 - NFDoubleCore.ToString Method 647
 - NFDoubleCore Properties 648
 - NFDoubleCore.Y Property 648
 - NFDoubleCore.X Property 648
- NFExtractor Class 648
 - NFExtractor Constructor 651
 - NFExtractor.NFExtractor Constructor () 651
 - NFExtractor.NFExtractor Constructor (bool) 651
 - NFExtractor Fields 651
 - NFExtractor.ModeAtmelFingerChip Field 651
 - NFExtractor.ModeAuthentecAes2501B Field 651
 - NFExtractor.ModeAuthentecAes4000 Field 652
 - NFExtractor.ModeAuthentecAFS2 Field 652

NFExtractor.ModeBiometrikaFX2000 Field	652
NFExtractor.ModeBiometrikaFX3000 Field	652
NFExtractor.ModeBmfBlp100 Field	652
NFExtractor.ModeCrossMatchVerifier300 Field	652
NFExtractor.ModeDigentlzzix Field	652
NFExtractor.ModeDigitalPersonaUareU Field	652
NFExtractor.ModeEthentica Field	652
NFExtractor.ModeFujitsuMbf200 Field	653
NFExtractor.ModeFutronicFS80 Field	653
NFExtractor.ModeGeneral Field	653
NFExtractor.ModeIdenticatorTechnologyDF90 Field	653
NFExtractor.ModeIdentixDfr2090 Field	653
NFExtractor.ModeIdentixTouchView Field	653
NFExtractor.ModeKeytronicSecureDesktop Field	653
NFExtractor.ModeLighTuningLttC500 Field	653
NFExtractor.ModeNitgenFingkeyHamster Field	653
NFExtractor.ModePreciseBiometrics100CS Field	654
NFExtractor.ModeSecugenHamster Field	654
NFExtractor.ModeStartekFM200 Field	654
NFExtractor.ModeTacomaCmos Field	654
NFExtractor.ModeTestechBiol Field	654
NFExtractor.ModeUpekTouchChip Field	654
NFExtractor.ParameterExtractedRidgeCounts Field	654
NFExtractor.ParameterGeneralizationMaximalRotation Field	654
NFExtractor.ParameterGeneralizationMinMinutiaCount Field	654
NFExtractor.ParameterGeneralizationThreshold Field	655
NFExtractor.ParameterMinMinutiaCount Field	655
NFExtractor.ParameterMode Field	655
NFExtractor.ParameterQualityThreshold Field	655
NFExtractor.ParameterReturnedImage Field	655
NFExtractor.ParameterTemplateSize Field	655
NFExtractor.ParameterUseQuality Field	655
NFExtractor Methods	655
NFExtractor.Extract Method	655
Generalize Method	656
NFExtractor.GetStaticParameter Method	656
NFExtractor.IsTemplateSmall Method	657
NFExtractor.MakeLightTemplate Method	657
NFExtractor.MakeSmallTemplate Method	657
NFExtractor.SetStaticParameter Method	658
NFExtractor.UpdateTemplate Method	658
NFExtractor Properties	658
NFExtractor.ExtractedRidgeCounts Property	658
NFExtractor.GeneralizationMaximalRotation Property	658

- NFExtractor.GeneralizationMinMinutiaCount Property 659
- NFExtractor.GeneralizationThreshold Property 659
- NFExtractor.IsPalm Property 659
- NFExtractor.MinMinutiaCount Property 659
- NFExtractor.Mode Property 659
- NFExtractor.QualityThreshold Property 660
- NFExtractor.ReturnedImage Property 660
- NFExtractor.TemplateSize Property 660
- NFExtractor.UseQuality Property 660
- NFMinutia Structure 660
 - NFMinutia Constructor 661
 - NFMinutia.NFMinutia Constructor (ushort, ushort, NFMinutiaType, byte) 661
 - NFMinutia.NFMinutia Constructor (ushort, ushort, NFMinutiaType, byte, byte, byte, byte) 661
 - NFMinutia.NFMinutia Constructor (ushort, ushort, NFMinutiaType, double) 662
 - NFMinutia.NFMinutia Constructor (ushort, ushort, NFMinutiaType, double, byte, byte, byte) 662
 - NFMinutia Methods 662
 - NFMinutia.ToString Method 662
 - NFMinutia Properties 662
 - NFMinutia.Angle Property 662
 - NFMinutia.Curvature Property 663
 - NFMinutia.G Property 663
 - NFMinutia.Y Property 663
 - NFMinutia.Quality Property 663
 - NFMinutia.RawAngle Property 664
 - NFMinutia.Type Property 664
 - NFMinutia.X Property 664
- NFMinutiaNeighbor Structure 664
 - NFMinutiaNeighbor.NFMinutiaNeighbor Constructor 665
 - NFMinutiaNeighbor Fields 665
 - NFMinutiaNeighbor.Empty Field 665
 - NFMinutiaNeighbor Methods 665
 - NFMinutiaNeighbor.ToString Method 665
 - NFMinutiaNeighbor Properties 665
 - NFMinutiaNeighbor.Index Property 665
 - NFMinutiaNeighbor.RidgeCount Property 666
- NFRecord Class 666
 - NFRecord Constructor 669
 - NFRecord.NFRecord Constructor (IntPtr, int) 669
 - NFRecord.NFRecord Constructor (IntPtr, int, out NFRecordInfo) 669
 - NFRecord.NFRecord Constructor (IntPtr, int, uint) 669
 - NFRecord.NFRecord Constructor (IntPtr, int, uint, out NFRecordInfo) 670
 - NFRecord.NFRecord Constructor (bool, ushort, ushort, ushort, ushort) 670
 - NFRecord.NFRecord Constructor (bool, ushort, ushort, ushort, ushort, uint) 670
 - NFRecord.NFRecord Constructor (byte[]) 671

NFRecord.NFRecord Constructor (byte[], out NFRecordInfo)	671
NFRecord.NFRecord Constructor (byte[], uint)	671
NFRecord.NFRecord Constructor (byte[], uint, out NFRecordInfo)	671
NFRecord.NFRecord Constructor (ushort, ushort, ushort, ushort)	672
NFRecord.NFRecord Constructor (ushort, ushort, ushort, ushort, uint)	672
NFRecord Classes	672
NFRecord.CoreCollection Class	672
NFRecord.DeltaCollection Class	673
NFRecord.DoubleCoreCollection Class	673
NFRecord.MinutiaCollection Class	673
NFRecord.MinutiaNeighborsCollection Class	673
NFRecord Fields	673
NFRecord.FlagSaveBlockedOrients Field	673
NFRecord.FlagSaveV2 Field	673
NFRecord.FlagSaveV3 Field	673
NFRecord.FlagSkipBlockedOrients Field	674
NFRecord.FlagSkipCurvatures Field	674
NFRecord.FlagSkipGs Field	674
NFRecord.FlagSkipQualities Field	674
NFRecord.FlagSkipRidgeCounts Field	674
NFRecord.FlagSkipSingularPoints Field	674
NFRecord.MaxCoreCount Field	674
NFRecord.MaxDeltaCount Field	674
NFRecord.MaxDimension Field	674
NFRecord.MaxDoubleCoreCount Field	675
NFRecord.MaxFingerCoreCount Field	675
NFRecord.MaxFingerDeltaCount Field	675
NFRecord.MaxFingerDimension Field	675
NFRecord.MaxFingerDoubleCoreCount Field	675
NFRecord.MaxFingerMinutiaCount Field	675
NFRecord.MaxMinutiaCount Field	675
NFRecord.MaxPalmCoreCount Field	675
NFRecord.MaxPalmDeltaCount Field	676
NFRecord.MaxPalmDimension Field	676
NFRecord.MaxPalmDoubleCoreCount Field	676
NFRecord.MaxPalmMinutiaCount Field	676
NFRecord.Resolution Field	676
NFRecord Methods	676
Check Method	676
NFRecord.Clone Method	677
FromHandle Method	677
GetCbeffProductType Method	677
GetG Method	678
GetHeight Method	679

- GetHorzResolution Method 679
- GetImpressionType Method 680
- GetMaxSize Method 680
- GetMaxSizeV1 Method 681
- GetPatternClass Method 682
- GetPosition Method 683
- GetQuality Method 684
- GetSize Method 684
- GetSizeV1 Method 685
- GetVertResolution Method 685
- GetWidth Method 686
- Save Method 687
- SaveV1 Method 689
- NFRecord.SortMinutiae Method 691
- NFRecord.TruncateMinutiae Method 691
- NFRecord.TruncateMinutiaeByQuality Method 691
- NFRecord Properties 691
 - NFRecord.CbeffProductType Property 691
 - NFRecord.Cores Property 692
 - NFRecord.Deltas Property 692
 - NFRecord.DoubleCores Property 692
 - NFRecord.G Property 692
 - NFRecord.Height Property 692
 - NFRecord.HorzResolution Property 692
 - NFRecord.ImpressionType Property 693
 - NFRecord.Minutiae Property 693
 - NFRecord.MinutiaeNeighbors Property 693
 - NFRecord.MinutiaFormat Property 693
 - NFRecord.PatternClass Property 693
 - NFRecord.Position Property 694
 - NFRecord.Quality Property 694
 - NFRecord.RidgeCountsType Property 694
 - NFRecord.VertResolution Property 694
 - NFRecord.Width Property 694
- NFRecordInfo Class 694
 - NFRecordInfo Properties 695
 - NFRecordInfo.HeaderSize Property 695
 - NFRecordInfo.MajorVersion Property 695
 - NFRecordInfo.MinorVersion Property 695
 - NFRecordInfo.Size Property 695
- NFTemplate Class 695
 - NFTemplate Constructor 697
 - NFTemplate.NFTemplate Constructor () 697
 - NFTemplate.NFTemplate Constructor (IntPtr, int) 697

NFTemplate.NFTemplate Constructor (IntPtr, int, out NFTemplateInfo)	697
NFTemplate.NFTemplate Constructor (IntPtr, int, uint)	697
NFTemplate.NFTemplate Constructor (IntPtr, int, uint, out NFTemplateInfo)	698
NFTemplate.NFTemplate Constructor (bool)	698
NFTemplate.NFTemplate Constructor (bool, uint)	699
NFTemplate.NFTemplate Constructor (byte[])	699
NFTemplate.NFTemplate Constructor (byte[], out NFTemplateInfo)	699
NFTemplate.NFTemplate Constructor (byte[], uint)	699
NFTemplate.NFTemplate Constructor (byte[], uint, out NFTemplateInfo)	700
NFTemplate.NFTemplate Constructor (uint)	700
NFTemplate Classes	701
NFTemplate.RecordCollection Class	701
NFTemplate Fields	702
NFTemplate.FlagProcessFirstRecordOnly Field	703
NFTemplate.MaxRecordCount Field	703
NFTemplate Methods	703
CalculateSize Method	703
Check Method	703
NFTemplate.Clone Method	704
FromHandle Method	704
GetRecordCount Method	705
GetSize Method	705
IsPalmTemplate Method	706
Pack Method	707
Save Method	708
Unpack Method	710
NFTemplate Properties	712
NFTemplate.IsPalm Property	712
NFTemplate.Records Property	712
NFTemplateInfo Class	712
NFTemplateInfo Classes	713
NFTemplateInfo.RecordInfoCollection Class	713
NFTemplateInfo Properties	713
NFTemplateInfo.HeaderSize Property	713
NFTemplateInfo.MajorVersion Property	713
NFTemplateInfo.MinorVersion Property	713
NFTemplateInfo.RecordInfos Property	713
NFTemplateInfo.Size Property	713
NleDetectionDetails Structure	714
NleDetectionDetails Properties	714
NleDetectionDetails.Eyes Property	714
NleDetectionDetails.EyesAvailable Property	714
NleDetectionDetails.Face Property	714
NleDetectionDetails.FaceAvailable Property	714

NleEyes Structure	714
NleEyes Properties	715
NleEyes.First Property	715
NleEyes.FirstConfidence Property	715
NleEyes.Second Property	715
NleEyes.SecondConfidence Property	715
NleFace Structure	715
NleFace Properties	716
NleFace.Confidence Property	716
NleFace.Rectangle Property	716
NleFace.Rotation Property	716
NleRotation Structure	716
NleRotation Properties	717
NleRotation.Yaw Property	717
NleRotation.Pitch Property	717
NleRotation.Roll Property	717
NLExtractor Class	717
NLExtractor.NLExtractor Constructor	719
NLExtractor Fields	719
NLExtractor.ParameterFaceConfidenceThreshold Field	719
NLExtractor.ParameterFaceQualityThreshold Field	719
NLExtractor.ParameterFavorLargestFace Field	720
NLExtractor.ParameterLivenessThreshold Field	720
NLExtractor.ParameterMaxIod Field	720
NLExtractor.ParameterMaxRecordsPerTemplate Field	720
NLExtractor.ParameterMaxRollAngleDeviation Field	720
NLExtractor.ParameterMinIod Field	720
NLExtractor.ParameterTemplateSize Field	720
NLExtractor.ParameterUseLivenessCheck Field	721
NLExtractor Methods	721
Compress Method	721
NLExtractor.DetectFace Method	721
NLExtractor.DetectFaces Method	722
NLExtractor.DetectFacialFeatures Method	722
NLExtractor.Extract Method	722
NLExtractor.ExtractNext Method	723
NLExtractor.ExtractStart Method	723
NLExtractor.ExtractUsingDetails Method	723
NLExtractor.Generalize Method	724
NLExtractor.GetStaticParameter Method	724
NLExtractor.SetStaticParameter Method	724
NLExtractor Properties	724
NLExtractor.FaceConfidenceThreshold Property	725
NLExtractor.FaceQualityThreshold Property	725

NLExtractor.FavorLargestFace Property	725
NLExtractor.LivenessThreshold Property	725
NLExtractor.Maxlod Property	725
NLExtractor.MaxRecordsPerTemplate Property	726
NLExtractor.MaxRollAngleDeviation Property	726
NLExtractor.Minlod Property	726
NLExtractor.TemplateSize Property	726
NLExtractor.UseLivenessCheck Property	726
NTemplate Class	726
NTemplate Constructor	728
NTemplate.NTemplate Constructor ()	728
NTemplate.NTemplate Constructor (IntPtr, int)	728
NTemplate.NTemplate Constructor (IntPtr, int, out NTemplateInfo)	728
NTemplate.NTemplate Constructor (IntPtr, int, uint)	728
NTemplate.NTemplate Constructor (IntPtr, int, uint, out NTemplateInfo)	729
NTemplate.NTemplate Constructor (byte[])	729
NTemplate.NTemplate Constructor (byte[], out NTemplateInfo)	729
NTemplate.NTemplate Constructor (byte[], uint)	729
NTemplate.NTemplate Constructor (byte[], uint, out NTemplateInfo)	730
NTemplate.NTemplate Constructor (uint)	730
NTemplate Methods	730
AddFaces Method	730
NTemplate.AddFacesCopy Method	731
AddFingers Method	731
NTemplate.AddFingersCopy Method	733
AddIrises Method	733
NTemplate.AddIrisesCopy Method	734
AddPalms Method	734
NTemplate.AddPalmsCopy Method	735
NTemplate.CalculateSize Method	735
Check Method	736
NTemplate.Clear Method	736
NTemplate.Clone Method	736
FromHandle Method	737
GetSize Method	737
Pack Method	738
NTemplate.RemoveFaces Method	739
NTemplate.RemoveFingers Method	739
NTemplate.RemoveIrises Method	739
NTemplate.RemovePalms Method	739
Save Method	739
Unpack Method	741
NTemplate Properties	743
NTemplate.Faces Property	743

NTemplate.Fingers Property	743
NTemplate.Irises Property	744
NTemplate.Palms Property	744
NTemplateInfo Class	744
NTemplateInfo Properties	744
NTemplateInfo.FacesInfo Property	745
NTemplateInfo.FingersInfo Property	745
NTemplateInfo.HeaderSize Property	745
NTemplateInfo.IrisesInfo Property	745
NTemplateInfo.MajorVersion Property	745
NTemplateInfo.MinorVersion Property	745
NTemplateInfo.PalmsInfo Property	745
NTemplateInfo.Size Property	745
NTemplates Class	745
NTemplates Fields	746
NTemplates.DllName Field	746
NTemplates Methods	746
NTemplates.GetInfo Method	746
6.2.2.2 Structs, Records, Enums	746
Neurotec.Biometrics.NBiometricType Enumeration	746
Neurotec.Biometrics.NfeExtractionStatus Enumeration	746
Neurotec.Biometrics.NfeReturnedImage Enumeration	747
Neurotec.Biometrics.NfeTemplateSize Enumeration	747
Neurotec.Biometrics.NFImpressionType Enumeration	747
Neurotec.Biometrics.NFMinutiaFormat Enumeration	748
Neurotec.Biometrics.NFMinutiaOrder Enumeration	749
Neurotec.Biometrics.NFMinutiaType Enumeration	749
Neurotec.Biometrics.NFPatternClass Enumeration	749
Neurotec.Biometrics.NFPosition Enumeration	750
Neurotec.Biometrics.NFRidgeCountsType Enumeration	752
Neurotec.Biometrics.NleExtractionStatus Enumeration	752
Neurotec.Biometrics.NleTemplateSize Enumeration	753
6.2.3 Neurotec.Biometrics.Standards Namespace	753
6.2.3.1 Classes	754
Bdif Class	754
Bdif Fields	755
Bdif.FlagDoNotCheckCbeffProductId Field	755
BdifFaceFeaturePoint Structure	755
BdifFaceFeaturePoint.BdifFaceFeaturePoint Constructor	755
BdifFaceFeaturePoint Methods	755
BdifFaceFeaturePoint.ToString Method	756
BdifFaceFeaturePoint Properties	756
BdifFaceFeaturePoint.Code Property	756
BdifFaceFeaturePoint.Y Property	756

BdifFaceFeaturePoint.Type Property	756
BdifFaceFeaturePoint.X Property	756
BdifFPMinutiaNeighbor Structure	756
BdifFPMinutiaNeighbor.BdifFPMinutiaNeighbor Constructor	757
BdifFPMinutiaNeighbor.Fields	757
BdifFPMinutiaNeighbor.Empty Field	757
BdifFPMinutiaNeighbor.Methods	757
BdifFPMinutiaNeighbor.ToString Method	757
BdifFPMinutiaNeighbor.Properties	758
BdifFPMinutiaNeighbor.Index Property	758
BdifFPMinutiaNeighbor.RidgeCount Property	758
CbeffBdbFormatIdentifiers Class	758
CbeffBdbFormatIdentifiers.Fields	766
CbeffBdbFormatIdentifiers.BioLogicaSistemasIso19794_2BioUniStation Field	766
CbeffBdbFormatIdentifiers.BioLogicaSistemasIso19794_4BioUniStation Field	766
CbeffBdbFormatIdentifiers.BioLogicaSistemasIso19794_5BioUniStation Field	766
CbeffBdbFormatIdentifiers.FederalOfficeForInformationSecurityTRBiometricsXmlFace10 Field	766
CbeffBdbFormatIdentifiers.FederalOfficeForInformationSecurityTRBiometricsXmlFinger10 Field	766
CbeffBdbFormatIdentifiers.Id3SemiconductorsCompactCardExtension Field	767
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsAviMedia Field	767
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsBiometricFusionData Field	767
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsBmplImage Field	767
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsEbts Field	767
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFacelImage Field	767
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerImage Field	767
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerMinutiaeN Field	767
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerMinutiaeU Field	768
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerMinutiaeX Field	768
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerPatternN Field	768
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsGenericFbiEftsRecord Field	768
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsGif89aImage Field	768
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsHandGeometryN Field	768
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsHandGeometryX Field	768
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsIrisPolar Field	768
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsIrisRectilinear Field	769
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsJpeg2000Image Field	769
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsJpegImage Field	769
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsMpeg1Part3Media Field	769
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsMpegMedia Field	769
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNiem Field	769
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistIrl1_2000Type10Record Field	769
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistIrl1_2000Type13Record Field	769
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistIrl1_2000Type14Record Field	770
CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistIrl1_2000Type15Record Field	770

CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2000Type16Record Field 770
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2000Type4Record Field 770
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2007Type10Record Field 770
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2007Type14Record Field 770
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2007Type17Record Field 770
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_200XRecordCollectionForTenPrintCapture Field 771
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsPngImage Field 771
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignCommonFeatureDataN Field 771
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignCommonFeatureDataX Field 771
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataN Field 771
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataX Field 771
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignRawDataN Field 771
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignRawDataX Field 772
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsTiffImage Field 772
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsVrml3DObjectData Field 772
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsWavAudio Field 772
 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsWsqImage Field 772
 CbeffBdbFormatIdentifiers.IsolecJtc1SC27ITSecurityTechniquesBiometricReferenceTemplateCertificate Field 772
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFacelImage Field 772
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerImage Field 772
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactN Field 773
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactRH Field 773
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactV Field 773
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactVH Field 773
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalN Field 773
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalRNH Field 773
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalV Field 773
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalVNH Field 773
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeRecordN Field 774
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeRecordX Field 774
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSkeletalDataCardCompact Field 774
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSkeletalDataCardNormal Field 774
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSkeletalDataRecord Field 774
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSpectralDiscreteFourierTransform Field 774
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSpectralGaborFilter Field 774
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSpectralQuantizedCOSinusoidalTriplet Field 775
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsHandGeometrySilhouette Field 775
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsIrisImagePolar Field 775
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsIrisImageRectilinear Field 775
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsSignatureSignTimeSeriesCompact Field 775
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsSignatureSignTimeSeriesFull Field 775
 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsVascularImageData Field 775
 CbeffBdbFormatIdentifiers.NeurotechnologyaNLRecord3 Field 776

CbeffBdbFormatIdentifiers.NeurotechnologyANTemplate Field	776
CbeffBiometricOrganizations Class	776
CbeffBiometricOrganizations Fields	779
CbeffBiometricOrganizations._123ID Field	779
CbeffBiometricOrganizations.A4Vision Field	779
CbeffBiometricOrganizations.AcSysBiometrics Field	779
CbeffBiometricOrganizations.ActivCard Field	779
CbeffBiometricOrganizations.AntheusTechnology Field	779
CbeffBiometricOrganizations.AOptix Field	779
CbeffBiometricOrganizations.Atmel Field	779
CbeffBiometricOrganizations.AuroraWirelessTechnologies Field	780
CbeffBiometricOrganizations.AuthenTec Field	780
CbeffBiometricOrganizations.Aware Field	780
CbeffBiometricOrganizations.BioKeyInternational Field	780
CbeffBiometricOrganizations.BioKeyInternational2 Field	780
CbeffBiometricOrganizations.BioLinkTechnologiesInternational Field	780
CbeffBiometricOrganizations.BioLogicaSistemas Field	780
CbeffBiometricOrganizations.Bioscrypt Field	780
CbeffBiometricOrganizations.Biovision Field	780
CbeffBiometricOrganizations.BioWise Field	781
CbeffBiometricOrganizations.CyberSign Field	781
CbeffBiometricOrganizations.CogentSystems Field	781
CbeffBiometricOrganizations.CryptoMetrics Field	781
CbeffBiometricOrganizations.CrossMatchTechnologies Field	781
CbeffBiometricOrganizations.DermalogIdentificationSystems Field	781
CbeffBiometricOrganizations.Digent Field	781
CbeffBiometricOrganizations.DigitalPersona Field	781
CbeffBiometricOrganizations.EasternGoldenFingerTechnologyBeijing Field	781
CbeffBiometricOrganizations.EastShoreTechnologies Field	782
CbeffBiometricOrganizations.ECryp Field	782
CbeffBiometricOrganizations.FederalOfficeForInformationSecurity Field	782
CbeffBiometricOrganizations.FingerprintCards Field	782
CbeffBiometricOrganizations.Fujitsu Field	782
CbeffBiometricOrganizations.GermanInstituteForStandardization Field	782
CbeffBiometricOrganizations.GreenBitAmericas Field	782
CbeffBiometricOrganizations.GriauleTecnologia Field	782
CbeffBiometricOrganizations.GuardwareSystems Field	782
CbeffBiometricOrganizations.Hitachi Field	783
CbeffBiometricOrganizations.HumanScan Field	783
CbeffBiometricOrganizations.Id3Semiconductors Field	783
CbeffBiometricOrganizations.Identix Field	783
CbeffBiometricOrganizations.Identix2 Field	783
CbeffBiometricOrganizations.ImageWareSystems Field	783
CbeffBiometricOrganizations.IncitsTCM1Biometrics Field	783

CbeffBiometricOrganizations.InfineonTechnologies Field 783
CbeffBiometricOrganizations.Innovatrics Field 783
CbeffBiometricOrganizations.InternationalBiometricGroup Field 784
CbeffBiometricOrganizations.InvestorsHelpers Field 784
CbeffBiometricOrganizations.IridianTechnologies Field 784
CbeffBiometricOrganizations.IsolecJtc1SC27ITSecurityTechniques Field 784
CbeffBiometricOrganizations.IsolecJtc1SC37Biometrics Field 784
CbeffBiometricOrganizations.JanusAssociates Field 784
CbeffBiometricOrganizations.LGElectronicsUsa Field 784
CbeffBiometricOrganizations.LogicoSmartcardSolutions Field 784
CbeffBiometricOrganizations.Lumidigm Field 784
CbeffBiometricOrganizations.NecSolutionsAmerica Field 785
CbeffBiometricOrganizations.Neurotechnologija Field 785
CbeffBiometricOrganizations.Nist Field 785
CbeffBiometricOrganizations.Nitgen Field 785
CbeffBiometricOrganizations.NotForUse Field 785
CbeffBiometricOrganizations.OmniMotorola Field 785
CbeffBiometricOrganizations.OmniPerception Field 785
CbeffBiometricOrganizations.PreciseBiometrics Field 785
CbeffBiometricOrganizations.Private Field 785
CbeffBiometricOrganizations.RecognitionSystems Field 786
CbeffBiometricOrganizations.Safink Field 786
CbeffBiometricOrganizations.SagemMorpho Field 786
CbeffBiometricOrganizations.SecuGen Field 786
CbeffBiometricOrganizations.SecureDesign Field 786
CbeffBiometricOrganizations.SilexTechnology Field 786
CbeffBiometricOrganizations.SondaTechnologies Field 786
CbeffBiometricOrganizations.StarTekEngineering Field 786
CbeffBiometricOrganizations.Suprema Field 786
CbeffBiometricOrganizations.Testing1 Field 787
CbeffBiometricOrganizations.Testing2 Field 787
CbeffBiometricOrganizations.ThalesIdenitfication Field 787
CbeffBiometricOrganizations.UltraScan Field 787
CbeffBiometricOrganizations.Upek Field 787
CbeffBiometricOrganizations.Validity Field 787
CbeffBiometricOrganizations.Veridicom Field 787
CbeffBiometricOrganizations.Veridt Field 787
CbeffBiometricOrganizations.Viisage Field 787
CbeffBiometricOrganizations.WarwickWarp Field 788
CbeffBiometricOrganizations.XTec Field 788
FMCARD Class 788
FMCARD Fields 788
FMCARD.DefaultMaxEnrollMC Field 788
FMCARD.DefaultMaxVerifyMC Field 789

FmCard.DefaultMinEnrollMC Field	789
FmCard.DefaultMinVerifyMC Field	789
FmCard Methods	789
ReadCompactSizeMinutiae Method	789
ReadNormalSizeMinutiae Method	791
WriteCompactSizeMinutiae Method	793
WriteNormalSizeMinutiae Method	797
FmcBdtTags Class	801
FmcBdtTags Fields	802
FmcBdtTags.CellQualityData Field	802
FmcBdtTags.CorePointData Field	802
FmcBdtTags.DeltaPointData Field	802
FmcBdtTags.FingerMinutiaeData Field	802
FmcBdtTags.RidgeCountData Field	802
FmcBitBhtBaptTags Class	802
FmcBitBhtBaptTags Fields	802
FmcBitBhtBaptTags.FeatureHandlingIndicator Field	803
FmcBitBhtBaptTags.MinutiaeOrder Field	803
FmcBitBhtBaptTags.NumberOfMinutiae Field	803
NBiometricStandards Class	803
NBiometricStandards Fields	803
NBiometricStandards.DllName Field	803
NBiometricStandards Methods	803
NBiometricStandards.GetInfo Method	803
6.2.3.2 Structs, Records, Enums	804
Neurotec.Biometrics.Standards.BdifEyeColor Enumeration	804
Neurotec.Biometrics.Standards.BdifEyePosition Enumeration	804
Neurotec.Biometrics.Standards.BdifFaceExpression Enumeration	805
Neurotec.Biometrics.Standards.BdifFaceFeaturePointType Enumeration	805
Neurotec.Biometrics.Standards.BdifFaceProperties Enumeration	805
Neurotec.Biometrics.Standards.BdifFPatternClass Enumeration	806
Neurotec.Biometrics.Standards.BdifFPImpressionType Enumeration	807
Neurotec.Biometrics.Standards.BdifFPMinutiaType Enumeration	808
Neurotec.Biometrics.Standards.BdifFPPosition Enumeration	809
Neurotec.Biometrics.Standards.BdifGender Enumeration	810
Neurotec.Biometrics.Standards.BdifHairColor Enumeration	811
Neurotec.Biometrics.Standards.BdifImageSourceType Enumeration	811
Neurotec.Biometrics.Standards.BdifIrisOrientation Enumeration	812
Neurotec.Biometrics.Standards.BdifIrisScanType Enumeration	812
Neurotec.Biometrics.Standards.BdifScaleUnits Enumeration	813
Neurotec.Biometrics.Standards.BdifStandard Enumeration	813
Neurotec.Biometrics.Standards.FmcFeatureHandling Enumeration	813
Neurotec.Biometrics.Standards.FmcMinutiaOrder Enumeration	814
6.2.4 Neurotec.DeviceManager Namespace	814

6.2.4.1 Classes 815

FPScanner Class 815

FPScanner Methods 816

FPScanner.StartCapturing Method 816

FPScanner.StartCapturingForOneImage Method 816

FPScanner.StopCapturing Method 816

FPScanner.ToString Method 817

FPScanner Properties 817

FPScanner.Id Property 817

FPScanner.IsCapturing Property 817

FPScanner.Owner Property 817

FPScanner Events 817

FPScanner.FingerPlaced Event 817

FPScanner.FingerRemoved Event 817

FPScanner.ImageScanned Event 818

FPScanner.IsCapturingChanged Event 818

FPScannerImageScannedEventArgs Class 818

FPScannerImageScannedEventArgs.FPScannerImageScannedEventArgs Constructor 818

FPScannerImageScannedEventArgs Properties 818

FPScannerImageScannedEventArgs.Image Property 818

FPScannerMan Class 819

FPScannerMan Constructor 820

FPScannerMan.FPScannerMan Constructor (ISynchronizeInvoke) 820

FPScannerMan.FPScannerMan Constructor (string, ISynchronizeInvoke) 820

FPScannerMan Classes 820

FPScannerMan.FPScannerCollection Class 820

FPScannerMan Methods 821

FPScannerMan.GetAvailableModules Method 821

FPScannerMan Properties 821

FPScannerMan.Scanners Property 821

FPScannerMan Events 821

FPScannerMan.ScannerAdded Event 821

FPScannerMan.ScannerRemoved Event 822

FPScannerManScannerEventArgs Class 822

FPScannerManScannerEventArgs.FPScannerManScannerEventArgs Constructor 822

FPScannerManScannerEventArgs Properties 822

FPScannerManScannerEventArgs.Scanner Property 822

NDeviceManager Class 822

NDeviceManager Fields 823

NDeviceManager.DllName Field 823

NDeviceManager Methods 823

NDeviceManager.GetInfo Method 823

6.2.4.2 Types 823

Neurotec.DeviceManager.FPScannerImageScannedEventHandler Type 823

Neurotec.DeviceManager.FPScannerManScannerEventHandler Type 824

6.2.5 Neurotec.Images Namespace 824

6.2.5.1 Classes 825

Bmp Class 825

Bmp Methods 825

LoadImage Method 825

LoadImageFromBitmap Method 826

Bmp.LoadImageFromHBitmap Method 827

SaveImage Method 827

Bmp.SaveImageToBitmap Method 828

Bmp.SaveImageToHBitmap Method 828

Jpeg Class 828

Jpeg Fields 829

Jpeg.DefaultQuality Field 829

Jpeg Methods 829

LoadImage Method 829

SaveImage Method 830

LosslessJpeg Class 831

LosslessJpeg Methods 832

SaveImage Method 832

NGrayscaleImage Class 832

NGrayscaleImage Methods 834

NGrayscaleImage.this Indexer 834

NImage Class 834

NImage Methods 835

NImage.Clone Method 835

NImage.Copy Method 836

Create Method 836

NImage.Crop Method 837

NImage.FlipDiagonally Method 838

NImage.FlipHorizontally Method 838

NImage.FlipVertically Method 838

NImage.FromBitmap Method 838

FromData Method 838

FromFile Method 843

FromHandle Method 844

NImage.FromHBitmap Method 844

FromImage Method 844

GetWrapper Method 848

NImage.RotateFlip Method 854

Save Method 854

NImage.ToBitmap Method 854

NImage.ToHBitmap Method 854

NImage Properties 855

- NImage.Height Property 855
- NImage.HorzResolution Property 855
- NImage.LongSize Property 855
- NImage.LongStride Property 855
- NImage.PixelFormat Property 856
- NImage.Pixels Property 856
- NImage.Size Property 856
- NImage.Stride Property 856
- NImage.VertResolution Property 857
- NImage.Width Property 857
- NImageFile Class 857
 - NImageFile Methods 858
 - NImageFile.Close Method 858
 - FromFile Method 858
 - NImageFile.ReadImage Method 858
 - NImageFile Properties 859
 - NImageFile.Format Property 859
- NImageFormat Class 859
 - NImageFormat Classes 860
 - NImageFormat.ImageFormatCollection Class 860
 - NImageFormat Fields 860
 - NImageFormat.Bmp Field 861
 - NImageFormat.Formats Field 861
 - NImageFormat.Gif Field 861
 - NImageFormat.IHead Field 861
 - NImageFormat.Jpeg Field 861
 - NImageFormat.Jpeg2K Field 861
 - NImageFormat.Png Field 861
 - NImageFormat.Tiff Field 861
 - NImageFormat.Wsq Field 861
 - NImageFormat Methods 862
 - LoadImage Method 862
 - OpenFile Method 863
 - SaveImage Method 864
 - SaveImages Method 864
 - NImageFormat.Select Method 865
 - NImageFormat Properties 865
 - NImageFormat.CanRead Property 866
 - NImageFormat.CanWrite Property 866
 - NImageFormat.CanWriteMultiple Property 866
 - NImageFormat.DefaultFileExtension Property 866
 - NImageFormat.FileFilter Property 866
 - NImageFormat.Name Property 866
- NImages Class 867

- NImages Fields 867
 - NImages.DllName Field 867
- NImages Methods 867
 - NImages.GetGrayscaleColorWrapper Method 867
 - NImages.GetInfo Method 868
 - NImages.GetOpenFileFilter Method 868
 - NImages.GetOpenFileFilterString Method 868
 - NImages.GetSaveFileFilter Method 868
 - NImages.GetSaveFileFilterString Method 868
- NMonochromeImage Class 869
 - NMonochromeImage Methods 870
 - NMonochromeImage.this Indexer 870
- NPixelFormat Structure 870
 - NPixelFormat Fields 871
 - NPixelFormat.Grayscale Field 871
 - NPixelFormat.Monochrome Field 871
 - NPixelFormat.Rgb Field 872
 - NPixelFormat Methods 872
 - CalcRowLongSize Method 872
 - CalcRowSize Method 872
 - NPixelFormat.Equals Method 873
 - NPixelFormat.GetHashCode Method 873
 - GetRowLongSize Method 873
 - GetRowSize Method 874
 - NPixelFormat.IsValid Method 875
 - NPixelFormat.ToString Method 875
 - NPixelFormat Operators 875
 - NPixelFormat.!= Operator 875
 - NPixelFormat.== Operator 875
 - NPixelFormat.NPixelFormat Operator 876
 - NPixelFormat.uint Operator 876
 - NPixelFormat Properties 876
 - NPixelFormat.BitsPerPixel Property 876
- NRgb Structure 876
 - NRgb.NRgb Constructor 877
 - NRgb Methods 877
 - NRgb.ToString Method 877
 - NRgb Properties 877
 - NRgb.Blue Property 877
 - NRgb.Green Property 877
 - NRgb.Red Property 878
- NRgbImage Class 878
 - NRgbImage Methods 879
 - NRgbImage.this Indexer 879

Png Class	880
Png Fields	880
Png.DefaultCompressionLevel Field	880
Png Methods	880
LoadImage Method	880
SaveImage Method	881
Tiff Class	883
Tiff Methods	883
LoadImage Method	883
6.2.5.2 Structs, Records, Enums	884
Neurotec.Images.BmpRecording Enumeration	884
Neurotec.Images.BmpSaveMode Enumeration	884
Neurotec.Images.NImageRotateFlipType Enumeration	885
6.2.6 Neurotec.IO Namespace	885
6.2.6.1 Classes	885
NBuffer Class	886
NBuffer.NBuffer Constructor	886
NBuffer Methods	886
NBuffer.ToArray Method	886
NBuffer.WriteTo Method	887
NBuffer Properties	887
NBuffer.Length Property	887
NBuffer.LongLength Property	887
NBuffer.Ptr Property	887
NBuffer.UIntPtrLength Property	887
NMemoryStream Class	887
NMemoryStream Constructor	888
NMemoryStream.NMemoryStream Constructor ()	888
NMemoryStream.NMemoryStream Constructor (long)	888
NMemoryStream.NMemoryStream Constructor (long, long)	889
NMemoryStream Methods	889
NMemoryStream.ToArray Method	889
NMemoryStream.WriteTo Method	889
NMemoryStream Properties	889
NMemoryStream.Capacity Property	889
NStream Class	889
NStream Methods	890
NStream.Close Method	890
NStream.Flush Method	890
FromHandle Method	890
NStream.FromStream Method	891
Read Method	891
NStream.ReadByte Method	892
NStream.Seek Method	892

NStream.SetLength Method	893
Write Method	893
NStream.WriteByte Method	894
NStream Properties	894
NStream.CanRead Property	894
NStream.CanSeek Property	894
NStream.CanWrite Property	894
NStream.Handle Property	894
NStream.Length Property	894
NStream.Position Property	894
6.2.6.2 Structs, Records, Enums	894
Neurotec.IO.NByteOrder Enumeration	895
6.2.7 Neurotec.Licensing Namespace	895
6.2.7.1 Classes	895
NLicense Class	895
NLicense Methods	895
NLicense.GetInfo Method	896
NLicense.IsComponentActivated Method	896
Obtain Method	896
NLicense.Release Method	897
NLicenseInfo Class	897
NLicenseInfo Properties	897
NLicenseInfo.DistributorId Property	897
NLicenseInfo.IsObtained Property	898
NLicenseInfo.SerialNumber Property	898
NLicensing Class	898
NLicensing Fields	898
NLicensing.DllName Field	898
NLicensing Methods	898
NLicensing.GetInfo Method	898
6.2.8 Neurotec.SmartCards Namespace	899
6.2.8.1 Classes	899
ApuClass Structure	899
ApuClass Constructor	900
ApuClass.ApuClass Constructor (bool, ApuClassSecureMessaging, int)	900
ApuClass.ApuClass Constructor (byte)	901
ApuClass Methods	901
ApuClass.Equals Method	901
ApuClass.GetHashCode Method	901
ApuClass.ToString Method	901
ApuClass Operators	901
ApuClass.!= Operator	901
ApuClass.== Operator	902
ApuClass.byte Operator	902

ApuClass Properties 902

- ApuClass.ChannelNumber Property 902
- ApuClass.IsInterindustry Property 902
- ApuClass.IsInterindustryFirst Property 903
- ApuClass.IsInterindustryFurther Property 903
- ApuClass.IsInterindustryReserved Property 903
- ApuClass.IsLastOrTheOnly Property 903
- ApuClass.IsProprietary Property 903
- ApuClass.IsValid Property 904
- ApuClass.SecureMessaging Property 904

ApuInstruction Structure 904

ApuInstruction Operator 907

- ApuInstruction.ApuInstruction Constructor (byte) 907
- ApuInstruction.ApuInstruction Operator (byte) 907

ApuInstruction Fields 908

- ApuInstruction.ActivateFile Field 908
- ApuInstruction.AppendRecord Field 908
- ApuInstruction.ChangeReferenceData Field 908
- ApuInstruction.CreateFile Field 908
- ApuInstruction.DeactivateFile Field 908
- ApuInstruction.DeleteFile Field 908
- ApuInstruction.DisableVerificationRequirement Field 908
- ApuInstruction.EnableVerificationRequirement Field 909
- ApuInstruction.Envelope Field 909
- ApuInstruction.EnvelopeBerTlv Field 909
- ApuInstruction.EraseBinary Field 909
- ApuInstruction.EraseBinaryBerTlv Field 909
- ApuInstruction.EraseRecord Field 909
- ApuInstruction.ExternalAuthenticate Field 909
- ApuInstruction.GeneralAuthenticate Field 909
- ApuInstruction.GeneralAuthenticateBerTlv Field 910
- ApuInstruction.GenerateAsymmetricKeyPair Field 910
- ApuInstruction.GetChallenge Field 910
- ApuInstruction.GetData Field 910
- ApuInstruction.GetDataBerTlv Field 910
- ApuInstruction.GetResponse Field 910
- ApuInstruction.InternalAuthenticate Field 910
- ApuInstruction.ManageChannel Field 910
- ApuInstruction.ManageSecurityEnvironment Field 911
- ApuInstruction.PerformScqlOperation Field 911
- ApuInstruction.PerformSecurityOperation Field 911
- ApuInstruction.PerformTransactionOperation Field 911
- ApuInstruction.PerformUserOperation Field 911
- ApuInstruction.PutData Field 911

- ApduInstruction.PutDataBerTlv Field 911
- ApduInstruction.ReadBinary Field 911
- ApduInstruction.ReadBinaryBerTlv Field 911
- ApduInstruction.ReadRecord Field 912
- ApduInstruction.ReadRecordBerTlv Field 912
- ApduInstruction.ResetRetryCounter Field 912
- ApduInstruction.SearchBinary Field 912
- ApduInstruction.SearchBinaryBerTlv Field 912
- ApduInstruction.SearchRecord Field 912
- ApduInstruction.Select Field 912
- ApduInstruction.TerminateCardUsage Field 912
- ApduInstruction.TerminateDF Field 913
- ApduInstruction.TerminateEF Field 913
- ApduInstruction.UpdateBinary Field 913
- ApduInstruction.UpdateBinaryBerTlv Field 913
- ApduInstruction.UpdateRecord Field 913
- ApduInstruction.UpdateRecordBerTlv Field 913
- ApduInstruction.Verify Field 913
- ApduInstruction.VerifyBerTlv Field 913
- ApduInstruction.WriteBinary Field 913
- ApduInstruction.WriteBinaryBerTlv Field 914
- ApduInstruction.WriteRecord Field 914
- ApduInstruction Methods 914
 - ApduInstruction.Equals Method 914
 - ApduInstruction.GetHashCode Method 914
 - ApduInstruction.ToString Method 914
- ApduInstruction Operators 914
 - ApduInstruction.!= Operator 915
 - ApduInstruction.== Operator 915
 - ApduInstruction.byte Operator 915
- ApduInstruction Properties 915
 - ApduInstruction.IsDataBerTlvEncoded Property 915
 - ApduInstruction.IsValid Property 916
- ApduStatus Structure 916
 - ApduStatus Operator 924
 - ApduStatus.ApduStatus Constructor (byte, byte) 924
 - ApduStatus.ApduStatus Constructor (ushort) 924
 - ApduStatus.ApduStatus Operator (ushort) 924
 - ApduStatus Fields 924
 - ApduStatus.DataStillAvailableXX Field 925
 - ApduStatus.ErrorClassNotSupported Field 925
 - ApduStatus.ErrorCnaAuthenticationMethodBlocked Field 925
 - ApduStatus.ErrorCnaCommandIncompatibleWithFileStructure Field 925
 - ApduStatus.ErrorCnaCommandNotAllowed Field 925

ApduStatus.ErrorCnaConditionsOfUseNotSatisfied Field 925
ApduStatus.ErrorCnaExpectedSecureMessagingDataObjectsMissing Field 925
ApduStatus.ErrorCnaIncorrectSecureMessagingDataObjects Field 925
ApduStatus.ErrorCnaNoInformation Field 926
ApduStatus.ErrorCnaReferenceDataNotUsable Field 926
ApduStatus.ErrorCnaSecurityStatusNotSatisfied Field 926
ApduStatus.ErrorCnaXX Field 926
ApduStatus.ErrorFicnsCommandChainingNotSupported Field 926
ApduStatus.ErrorFicnsLastCommandOfTheChainExpected Field 926
ApduStatus.ErrorFicnsLogicalChannelNotSupported Field 926
ApduStatus.ErrorFicnsNoInformation Field 927
ApduStatus.ErrorFicnsSecureMessagingNotSupported Field 927
ApduStatus.ErrorFicnsXX Field 927
ApduStatus.ErrorInstructionNotSupportedOrInvalid Field 927
ApduStatus.ErrorNoPreciseDiagnosis Field 927
ApduStatus.ErrorProprietary67XX Field 927
ApduStatus.ErrorProprietary6BXX Field 927
ApduStatus.ErrorProprietary6DXX Field 927
ApduStatus.ErrorProprietary6EXX Field 928
ApduStatus.ErrorProprietary6FXX Field 928
ApduStatus.ErrorSecurityXX Field 928
ApduStatus.ErrorSonvmcMemoryFailure Field 928
ApduStatus.ErrorSonvmcNoInformation Field 928
ApduStatus.ErrorSonvmcXX Field 928
ApduStatus.ErrorSonvmuExecutionError Field 928
ApduStatus.ErrorSonvmuImmediateResponseRequiredByTheCard Field 928
ApduStatus.ErrorSonvmuTriggeringByTheCardEnd Field 929
ApduStatus.ErrorSonvmuTriggeringByTheCardStart Field 929
ApduStatus.ErrorSonvmuXX Field 929
ApduStatus.ErrorWP1P2DFNameAlreadyExists Field 929
ApduStatus.ErrorWP1P2FileAlreadyExists Field 929
ApduStatus.ErrorWP1P2FileOrApplicationNotFound Field 929
ApduStatus.ErrorWP1P2FunctionNotSupported Field 929
ApduStatus.ErrorWP1P2IncorrectParametersInTheCommandDataField Field 929
ApduStatus.ErrorWP1P2IncorrectParametersP1P2 Field 930
ApduStatus.ErrorWP1P2NclInconsistentWithParametersP1P2 Field 930
ApduStatus.ErrorWP1P2NclInconsistentWithTlvStructure Field 930
ApduStatus.ErrorWP1P2NoInformation Field 930
ApduStatus.ErrorWP1P2NoQualification Field 930
ApduStatus.ErrorWP1P2NotEnoughMemorySpaceInTheFile Field 930
ApduStatus.ErrorWP1P2RecordNotFound Field 930
ApduStatus.ErrorWP1P2ReferencedDataOrReferenceDataNotFound Field 931
ApduStatus.ErrorWP1P2XX Field 931
ApduStatus.ErrorWrongLength Field 931

- ApduStatus.ErrorWrongLeXX Field 931
- ApduStatus.NoQualification Field 931
- ApduStatus.Proprietary9XXX Field 931
- ApduStatus.WarningSonvmcCounterX Field 931
- ApduStatus.WarningSonvmcFileFilledUpByTheLastWrite Field 931
- ApduStatus.WarningSonvmcNoInformation Field 932
- ApduStatus.WarningSonvmcXX Field 932
- ApduStatus.WarningSonvmuEndOfFileOrRecordBeforeReadingNeBytes Field 932
- ApduStatus.WarningSonvmuFileControlInformationIncorrectlyFormatted Field 932
- ApduStatus.WarningSonvmuNoInformation Field 932
- ApduStatus.WarningSonvmuNoInputDataAvailableFromASensorOnTheCard Field 932
- ApduStatus.WarningSonvmuPartOfReturnedDataMayBeCorrupted Field 932
- ApduStatus.WarningSonvmuSelectedFileDeactivated Field 933
- ApduStatus.WarningSonvmuSelectedFileInTerminationState Field 933
- ApduStatus.WarningSonvmuTriggeringByTheCardEnd Field 933
- ApduStatus.WarningSonvmuTriggeringByTheCardStart Field 933
- ApduStatus.WarningSonvmuXX Field 933
- ApduStatus Methods 933
 - ApduStatus.Equals Method 933
 - ApduStatus.GetHashCode Method 933
 - ApduStatus.ToString Method 934
- ApduStatus Operators 934
 - ApduStatus.!= Operator 934
 - ApduStatus.== Operator 934
 - ApduStatus.ushort Operator 934
- ApduStatus Properties 935
 - ApduStatus.DataStillAvailableValue Property 935
 - ApduStatus.ErrorSonvmuTriggeringByTheCardValue Property 935
 - ApduStatus.ErrorWrongLeValue Property 935
 - ApduStatus.IsCheckingError Property 935
 - ApduStatus.IsDataStillAvailable Property 935
 - ApduStatus.IsErrorCna Property 936
 - ApduStatus.IsErrorFicns Property 936
 - ApduStatus.IsErrorProprietary67XX Property 936
 - ApduStatus.IsErrorProprietary6BXX Property 936
 - ApduStatus.IsErrorProprietary6DXX Property 936
 - ApduStatus.IsErrorProprietary6EXX Property 937
 - ApduStatus.IsErrorProprietary6FXX Property 937
 - ApduStatus.IsErrorSecurity Property 937
 - ApduStatus.IsErrorSonvmc Property 937
 - ApduStatus.IsErrorSonvmu Property 937
 - ApduStatus.IsErrorSonvmuTriggeringByTheCard Property 937
 - ApduStatus.IsErrorWP1P2 Property 938
 - ApduStatus.IsErrorWrongLe Property 938

- ApduStatus.IsExecutionError Property 938
- ApduStatus.IsInterindustry Property 938
- ApduStatus.IsNormalProcessing Property 938
- ApduStatus.IsProcessAborted Property 939
- ApduStatus.IsProcessComplete Property 939
- ApduStatus.IsProprietary Property 939
- ApduStatus.IsProprietary9XXX Property 939
- ApduStatus.IsStateOfNonVolatileMemoryChanged Property 939
- ApduStatus.IsStateOfNonVolatileMemoryUnchanged Property 939
- ApduStatus.IsValid Property 940
- ApduStatus.IsWarningProcessing Property 940
- ApduStatus.IsWarningSonvmc Property 940
- ApduStatus.IsWarningSonvmcCounter Property 940
- ApduStatus.IsWarningSonvmu Property 940
- ApduStatus.IsWarningSonvmuTriggeringByTheCard Property 941
- ApduStatus.SW1 Property 941
- ApduStatus.SW2 Property 941
- ApduStatus.WarningSonvmcCounterValue Property 941
- ApduStatus.WarningSonvmuTriggeringByTheCardValue Property 941
- BerTag Structure 941
 - BerTag Operator 944
 - BerTag.BerTag Constructor (BerTagClass, BerTagEncoding, int) 944
 - BerTag.BerTag Constructor (int) 944
 - BerTag.BerTag Operator (int) 944
 - BerTag Fields 945
 - BerTag.BmpString Field 945
 - BerTag.Boolean Field 945
 - BerTag.CharacterString Field 945
 - BerTag.EmbeddedPdv Field 945
 - BerTag.Enumerated Field 945
 - BerTag.External Field 945
 - BerTag.GeneralizedTime Field 945
 - BerTag.GeneralString Field 945
 - BerTag.GraphicString Field 946
 - BerTag.IA5String Field 946
 - BerTag.InstanceOf Field 946
 - BerTag.Integer Field 946
 - BerTag.Iso646String Field 946
 - BerTag.Null Field 946
 - BerTag.NumericString Field 946
 - BerTag.ObjectDescriptor Field 946
 - BerTag.ObjectIdentifier Field 946
 - BerTag.OctetString Field 947
 - BerTag.PrintableString Field 947

- BerTag.Real Field 947
- BerTag.RelativeObjectIdentifier Field 947
- BerTag.Sequence Field 947
- BerTag.SequenceOf Field 947
- BerTag.Set Field 947
- BerTag.SetOf Field 947
- BerTag.T61String Field 947
- BerTag.TeletexString Field 948
- BerTag.UniversalString Field 948
- BerTag.UtcTime Field 948
- BerTag.Utf8String Field 948
- BerTag.VideotexString Field 948
- BerTag.VisibleString Field 948
- BerTag Methods 948
 - BerTag.Equals Method 948
 - BerTag.GetHashCode Method 949
 - BerTag.IsValid Method 949
 - Read Method 949
 - BerTag.Save Method 950
 - BerTag.ToString Method 950
 - Write Method 950
- BerTag Operators 951
 - BerTag.!= Operator 951
 - BerTag.== Operator 951
 - BerTag.int Operator 951
- BerTag Properties 952
 - BerTag.Class Property 952
 - BerTag.Encoding Property 952
 - BerTag.Length Property 952
 - BerTag.Number Property 952
- BerTlv Class 952
 - BerTlv Methods 953
 - BerTlv.Clone Method 953
 - Create Method 954
 - FromHandle Method 955
 - FromMemory Method 955
 - BerTlv.GetSizeMany Method 957
 - LoadMany Method 957
 - Save Method 959
 - SaveMany Method 960
 - BerTlv Properties 961
 - BerTlv.Length Property 961
 - BerTlv.MinLengthSize Property 961
 - BerTlv.Size Property 961

BerTlv.Tag Property	961
ConstructedBerTlv Class	961
ConstructedBerTlv Classes	963
ConstructedBerTlv.DataObjectCollection Class	963
ConstructedBerTlv Properties	966
ConstructedBerTlv.DataObjects Property	966
PrimitiveBerTlv Class	967
PrimitiveBerTlv Methods	968
GetValue Method	968
PrimitiveBerTlv.GetValueAsByte Method	968
PrimitiveBerTlv.GetValueAsBoolean Method	969
PrimitiveBerTlv.GetValueAsInt16 Method	969
PrimitiveBerTlv.GetValueAsInt32 Method	969
PrimitiveBerTlv.GetValueAsInteger Method	969
PrimitiveBerTlv.GetValueAsSByte Method	969
PrimitiveBerTlv.GetValueAsUInt16 Method	970
PrimitiveBerTlv.GetValueAsUInt32 Method	970
PrimitiveBerTlv.SetLength Method	970
SetValue Method	970
PrimitiveBerTlv.SetValueAsInteger Method	972
PrimitiveBerTlv Properties	972
PrimitiveBerTlv.ValuePtr Property	972
SCardTags Class	972
SCardTags Fields	977
SCardTags.AccountType Field	977
SCardTags.Address Field	977
SCardTags.AnswerToReset Field	977
SCardTags.ApplicationEffectiveDate Field	977
SCardTags.ApplicationExpirationDate Field	978
SCardTags.ApplicationFamilyIdentifier Field	978
SCardTags.ApplicationIdentifier Field	978
SCardTags.ApplicationImage Field	978
SCardTags.ApplicationImageTemplate Field	978
SCardTags.ApplicationLabel Field	978
SCardTags.ApplicationRelatedData Field	978
SCardTags.ApplicationTemplate Field	978
SCardTags.ApplicationTrack1 Field	978
SCardTags.ApplicationTrack2 Field	979
SCardTags.ApplicationTrack3 Field	979
SCardTags.AuthenticationData Field	979
SCardTags.BankIdentifierCode Field	979
SCardTags.BiometricDataTemplate Field	979
SCardTags.BiometricInformationGroupTemplate Field	979
SCardTags.BiometricInformationTemplate Field	979

SCardTags.CardCapabilities Field 979
SCardTags.CardData Field 980
SCardTags.CardEffectiveDate Field 980
SCardTags.CardExpirationDate Field 980
SCardTags.CardholderBiometricData Field 980
SCardTags.CardholderCertificate Field 980
SCardTags.CardholderHandwrittenSignatureImage Field 980
SCardTags.CardholderImageTemplate Field 980
SCardTags.CardholderName Field 980
SCardTags.CardholderNationality Field 980
SCardTags.CardholderPortraitImage Field 981
SCardTags.CardholderPrivateKey Field 981
SCardTags.CardholderPrivateKeyTemplate Field 981
SCardTags.CardholderPublicKey Field 981
SCardTags.CardholderPublicKeyTemplate Field 981
SCardTags.CardholderRelatedData Field 981
SCardTags.CardholderRequirementsExcludedFeatures Field 981
SCardTags.CardholderRequirementsIncludedFeatures Field 981
SCardTags.CardIssuersData Field 981
SCardTags.CardSequenceNumber Field 982
SCardTags.CardServiceData Field 982
SCardTags.CardTrack1 Field 982
SCardTags.CardTrack2 Field 982
SCardTags.CardTrack3 Field 982
SCardTags.CertificateContent Field 982
SCardTags.CertificateContentTemplate Field 982
SCardTags.CertificateHolderAuthorization Field 982
SCardTags.CoexistentTagAllocationAuthority Field 983
SCardTags.CommandToPerform Field 983
SCardTags.CompatibleTagAllocationAuthority Field 983
SCardTags.CountryCode Field 983
SCardTags.CountryCodeAlpha2Format Field 983
SCardTags.CountryCodeAlpha3Format Field 983
SCardTags.CountryCodeAndNationalData Field 983
SCardTags.CurrencyCode Field 983
SCardTags.CurrencyExponent Field 983
SCardTags.DateOfBirth Field 984
SCardTags.Deprecated Field 984
SCardTags.DigitalSignature Field 984
SCardTags.DigitalSignatureBlock Field 984
SCardTags.DiscretionaryData Field 984
SCardTags.DiscretionaryDataObjects Field 984
SCardTags.DisplayControl Field 984
SCardTags.DisplayMessage Field 984

SCardTags.DynamicAuthenticationTemplate Field	984
SCardTags.DynamicExternalAuthentication Field	985
SCardTags.DynamicInternalAuthentication Field	985
SCardTags.DynamicMutualAuthentication Field	985
SCardTags.ElementList Field	985
SCardTags.ExtendedHeaderList Field	985
SCardTags.FciTemplate Field	985
SCardTags.FcpTemplate Field	985
SCardTags.FileReference Field	985
SCardTags.FmdTemplate Field	985
SCardTags.HeaderList Field	986
SCardTags.HistoricalBytes Field	986
SCardTags.InitialAccessData Field	986
SCardTags.IntegratedCircuitManufacturerIdentifier Field	986
SCardTags.InterchangeControl Field	986
SCardTags.InterchangeProfile Field	986
SCardTags.InterindustryDataObjects Field	986
SCardTags.InternationalBankAccountNumber Field	986
SCardTags.IssuerIdentificationNumber Field	986
SCardTags.LanguagePreferences Field	987
SCardTags.LoginDataProprietary Field	987
SCardTags.LoginTemplate Field	987
SCardTags.MessageReference Field	987
SCardTags.Name Field	987
SCardTags.NonInterindustryDataObjects70 Field	987
SCardTags.NonInterindustryDataObjects71 Field	987
SCardTags.NonInterindustryDataObjects72 Field	987
SCardTags.NonInterindustryDataObjects74 Field	988
SCardTags.NonInterindustryDataObjects75 Field	988
SCardTags.NonInterindustryDataObjects76 Field	988
SCardTags.NonInterindustryDataObjects77 Field	988
SCardTags.OffsetDataObject Field	988
SCardTags.PinUsagePolicy Field	988
SCardTags.PrelIssuingData Field	988
SCardTags.PrimaryAccountNumber Field	988
SCardTags.PublicKeyOfCertificationAuthority Field	988
SCardTags.QualifiedName Field	989
SCardTags.SecureMessagingTemplate Field	989
SCardTags.SecurityEnvironmentTemplate Field	989
SCardTags.SecuritySupportTemplate Field	989
SCardTags.ServiceCode Field	989
SCardTags.Sex Field	989
SCardTags.SpecialUserRequirements Field	989
SCardTags.StaticInternalAuthentication Field	989

- SCardTags.StaticInternalAuthenticationFirstAssociatedData Field 989
- SCardTags.StaticInternalAuthenticationSecondAssociatedData Field 990
- SCardTags.StatusInformation Field 990
- SCardTags.TagList Field 990
- SCardTags.Timer Field 990
- SCardTags.TransactionCounter Field 990
- SCardTags.TransactionDate Field 990
- SCardTags.UniformResourceLocator Field 990
- SCardTags.Wrapper Field 990
- SimpleTag Structure 991
 - SimpleTag Operator 991
 - SimpleTag.SimpleTag Constructor (byte) 991
 - SimpleTag.SimpleTag Operator (byte) 991
 - SimpleTag Methods 992
 - SimpleTag.Equals Method 992
 - SimpleTag.GetHashCode Method 992
 - SimpleTag.ToString Method 992
 - SimpleTag Operators 992
 - SimpleTag.!= Operator 992
 - SimpleTag.== Operator 993
 - SimpleTag.byte Operator 993
 - SimpleTag Properties 993
 - SimpleTag.IsValid Property 993
- SimpleTlv Class 993
 - SimpleTlv Constructor 994
 - SimpleTlv.SimpleTlv Constructor (IntPtr, int) 995
 - SimpleTlv.SimpleTlv Constructor (SimpleTag) 995
 - SimpleTlv.SimpleTlv Constructor (SimpleTag, bool) 995
 - SimpleTlv.SimpleTlv Constructor (byte[]) 995
 - SimpleTlv.SimpleTlv Constructor (byte[], int, int) 995
 - SimpleTlv Methods 996
 - SimpleTlv.Clone Method 996
 - FromHandle Method 996
 - SimpleTlv.GetSizeMany Method 996
 - GetValue Method 997
 - LoadMany Method 997
 - Save Method 998
 - SaveMany Method 999
 - SimpleTlv.SetLength Method 999
 - SetValue Method 1000
 - SimpleTlv Properties 1000
 - SimpleTlv.Length Property 1000
 - SimpleTlv.Size Property 1000
 - SimpleTlv.Tag Property 1000

SimpleTlv.UseThreeByteLengthAlways Property	1001
SimpleTlv.ValuePtr Property	1001
6.2.8.2 Structs, Records, Enums	1001
Neurotec.SmartCards.ApuClassSecureMessaging Enumeration	1001
Neurotec.SmartCards.BerTagClass Enumeration	1001
Neurotec.SmartCards.BerTagEncoding Enumeration	1002
6.2.9 Neurotec.SmartCards.Biometry Namespace	1002
6.2.9.1 Classes	1003
SCardBdtCTTags Class	1003
SCardBdtCTTags Fields	1003
SCardBdtCTTags.Challenge Field	1003
SCardBdtCTTags.ChallengeQualifier Field	1003
SCardBdtTags Class	1003
SCardBdtTags Fields	1004
SCardBdtTags.ChallengeData Field	1004
SCardBdtTags.ChallengeTemplate Field	1004
SCardBdtTags.ProprietaryBiometricData Field	1004
SCardBdtTags.ProprietaryBiometricDataObjects Field	1004
SCardBdtTags.StandardBiometricData Field	1004
SCardBdtTags.StandardBiometricDataObjects Field	1005
SCardBitBhtTags Class	1005
SCardBitBhtTags Fields	1006
SCardBitBhtTags.BiometricAlgorithmParametersData Field	1006
SCardBitBhtTags.BiometricAlgorithmParametersTemplate Field	1006
SCardBitBhtTags.BiometricSubtype Field	1006
SCardBitBhtTags.BiometricType Field	1006
SCardBitBhtTags.CreationDate Field	1006
SCardBitBhtTags.Creator Field	1006
SCardBitBhtTags.FormatOwner Field	1006
SCardBitBhtTags.FormatType Field	1007
SCardBitBhtTags.Index Field	1007
SCardBitBhtTags.PatronHeaderVersion Field	1007
SCardBitBhtTags.ProductIdentifier Field	1007
SCardBitBhtTags.ProprietaryBht Field	1007
SCardBitBhtTags.StandardBht Field	1007
SCardBitBhtTags.ValidityPeriod Field	1007
SCardBitTags Class	1007
SCardBitTags Fields	1008
SCardBitTags.AlgorithmReference Field	1008
SCardBitTags.BiometricHeaderTemplate Field	1008
SCardBitTags.ReferenceDataQualifier Field	1008
SCardBitTags.StandardDataObjects Field	1008
SCardBitTags.TagAllocationAuthorityDataObjects Field	1008
6.2.9.2 Structs, Records, Enums	1009

Neurotec.SmartCards.Biometry.SCardBiometricSubtype Enumeration 1009

Neurotec.SmartCards.Biometry.SCardBiometricType Enumeration 1009

Neurotec.SmartCards.Biometry.SCardChallengeQualifier Enumeration 1010

6.3 Change Log 1010

6.3.1 C/C++ API Reference 1011

6.3.1.1 NBiometricStandards Library 1011

6.3.1.2 NCore Library 1014

6.3.1.3 NDeviceManager Library 1016

6.3.1.4 NExtractors Library 1016

6.3.1.5 NImages Library 1019

6.3.1.6 NLicensing Library 1022

6.3.1.7 NSmartCards Library 1022

6.3.1.8 NTemplates Library 1022

6.3.2 .NET API Reference 1024

6.3.2.1 Neurotec Library 1024

6.3.2.2 Neurotec.Biometric.Standards 1026

6.3.2.3 Neurotec.Biometrics.Extractors 1027

6.3.2.4 Neurotec.Biometrics.Templates 1027

6.3.2.5 Neurotec.DeviceManager 1028

6.3.2.6 Neurotec.Images 1028

6.3.2.7 Neurotec.Licensing 1030

6.3.2.8 Neurotec.SmartCards Library 1030

7 Support 1032

Index a

1 About

MegaMatcher On Card 2.0 SDK allows to perform enrollment and verification of fingerprint and / or face biometric templates on a Java Card. It also allows to integrate fingerprint and / or face templates storage and verification on a Java Card facilities to existing biometric systems based on *MegaMatcher SDK* or *VeriffFinger SDK* and / or *VeriLook SDK*. Both the Java Card and PC part components of the *MegaMatcher On Card SDK* are developed utilizing a set of ISO/IEC standards (7816-3:2006, 7816-4:2005, 7816-11:2004 and 19794-2:2005) enabling interoperability with and easy integration into existing smart card and/or biometrical systems. This enables the usage of MegaMatcher On Card fingerprint matching on-card engine within biometric systems using other vendors ISO/IEC fingerprint minutiae card formats compatible extractors.

1.1 Supported Platforms

The Java Card side components of the MegaMatcher On Card 2.0 SDK support smartcards implementing GlobalPlatform 2.1.1 and Java Card 2.2.1 or Java Card 2.2.2 specifications.

The PC part of the MegaMatcher On Card 2.0 SDK supports Windows and Linux platforms based on x86 or x64 processor architecture. Libraries for Windows and Linux operating systems are provided. .NET wrappers of Windows libraries are provided for .NET developers.

1.2 System Requirements

System requirements for installation and usage of components on JavaCard:

- GlobalPlatform 2.1.1 / Open Platform 2.0.1 and Java Card 2.2.1 / 2.2.2 specifications compatible smartcards for native / pure Java level on-card matching engine library usage.
- Pure native OS smartcards for customized native on-card matching engine library usage (please contact Neurotechnology for possible customizations).
- Transient (RAM) and persistent (EEPROM) memory requirements depend on the target (pure native / Java Card) smartcard platform and the implementation level (native / Java) of the matching engine library chosen as well as on the performance requirements. Different configurations of the on-card matching engine library are available for different:
 - implementation level,
 - speed,
 - accuracy and
 - memory consumption.

Please, contact Neurotechnology for details. Contacts are provided in section Support (see page 1032).

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

- PC with x86 or x64 compatible CPU
- Windows 9x/ME/2000/2003/XP/Vista/7 OS
- Microsoft .NET framework 2.0 (for .NET components) or newer

- Microsoft Visual Studio 2005 SP1 or newer (for application development)

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

- Computer with x86 compatible CPU
- Linux OS 2.4 or newer
- GCC-3.3.x or newer
- GNU Make 3.80 or newer
- PCSC-Lite 1.4.4 or newer
- ccid-1.3.0 or newer

1.3 MegaMatcher OnCard Structure

The MegaMatcher On Card 2.0 SDK contains the following components:

- PC-side software including MegaMatcher On Card Client Extractor for smartcard compatible templates extraction and sending those to a card.
- Java Card side software including applications using on-card biometric engine library for smartcard compatible templates (received from the PC-side) enrollment and verification within a smartcard.

The Native / Java compiled code of the engine is not included in this version. 3 smartcards with preloaded fingerprint and / or face matching engine are included for evaluation purposes. (Those may be provided by customer to test the library on the target specific project platform.) Additional smartcards with preloaded engine are also available.

To use on-card matching engine on a wide number of cards:

- the natively integrated version of biometric templates on-card matching engine or
- compiled pure Java code version of biometric templates on-card matching engine (that can be loaded into smartcards by an integrator) are also available. Please, contact Neurotechnology for details.

1.4 Licensing

MegaMatcher On Card 2.0 SDK includes:

- 2 MegaMatcher On Card fingerprint extractor licenses
- 2 MegaMatcher On Card face extractor licenses
- 3 smartcards with the preloaded fingerprint and / or face matching engine.

Integrators can develop only an **end-user** product using MegaMatcher On Card SDK and sell/install the product to their own customers. If the integrator wants to develop and sell a MegaMatcher On Card based development tool (with API, programming possibilities, programming samples, etc.), he/she will need a permission from Neurotechnology and shall sign a special VAR agreement.

Customers should obtain additional licenses for MegaMatcher On Card 2.0 SDK in order to use the product.

Three types of licensing for fingerprint and face Extractor of MegaMatcher On Card 2.0 SDK are available:

- **Using existing licenses.** Existing MegaMatcher, VeriFinger and VeriLook SDKs customers can use purchased Extractor licenses. MegaMatcher SDK licenses can be used with both fingers and faces extractors for MegaMatcher On Card 2.0 SDK. VeriFinger licenses can be used only with fingers extractors and VeriLook licenses - only with faces extractors.
- **Using MegaMatcher On Card 2.0 SDK licenses.** If customer does not use any of Neurotechnology SDKs

(MegaMatcher, VeriLok or VeriFinger), he can obtain fingerprint or face Extractor license for MegaMatcher On Card 2.0 SDK.

- **Using with other Extractor.** Also MegaMatcher On Card 2.0 SDK can be used with other fingerprint Extractor which supports ISO/IEC 19794-2:2005 compact size finger minutiae card standard format.

Additional cards with installed fingerprint and/or face matching component for the product development and deployment can be obtained at any time from Neurotechnology or a third party vendor, according to stock and availability.

For **large projects** that include more than 100,000 card licenses, MegaMatcher On Card matching components are available with different forms of licensing. Please contact us (info@neurotechnology.com) for more information.

1.4.1 Single computer license

A single computer license allows to install and run a MegaMatcher On Card Fingerprint Extractor or Face Extractor component installation on a single Personal Computer. The component license will not be lost if computer will be reinstalled.

The following license management options are available:

- license activation online by communicating with Neurotechnology's server;
- license activation by email;
- license activation using volume license manager;

Single computer license activated over Internet or by email is not suitable for virtual environments. Volume license manager used as a dongle would be required.

2 Overview

The MegaMatcher On Card 2.0 SDK is divided into the following parts:

- On-Card Biometric APIs (see page 4) (JavaCard part). Contains on-card matching engine implementation documentation also packages and sample applets that should be uploaded and installed respectively on a Java Card to enable face and / or fingerprint enrollment and verification.
- PC Part (see page 5). Contains libraries for smartcard compatible templates extraction, conversion and communication with a smartcard. Also samples and tutorials showing how to perform face and / or fingerprint templates enrollment and verification on a smartcard are presented.

2.1 On-Card Biometric Components

On-card components reside in the /JavaCard directory that is divided in several subdirectories containing:

- On-card biometric templates matching library API files
- On-card biometric server (manager) and biometric client sample applets demonstrating the use of on-card matching library API.

As the on-card biometric matching engine library implements Biometric Application Programming Interface (API) for Java Card Prepared by NIST/Biometric Consortium Biometric Interoperability, Assurance, and Performance Working Group, version 1.1, the org.javacardforum.javacard.biometry package stub files are not distributed and should be downloaded from Java Card Forum website (see http://javacardforum.org/03_documents/00_documents/fileload_03.zip for details and run the /JavaCard/BuildJCBioAPI.bat on Windows platform or JavaCard/BuildJCBioAPI.sh script on Linux platforms respectively). The on-card face and / or fingerprint templates matching library resides within the /JavaCard/JCBioAPI folder and includes the following files:

- Library stub class and export (EXP) files indispensable respectively for the compilation and further conversion of the Java Card packages that use the aforementioned biometry library (see /JavaCard/JCBioAPI/bio11 for details).
- Additionally stub source files and Javadoc of the fingerprint and / or face matcher library
org.javacardforum.javacard.biometry API is provided (see /JavaCard/JCBioAPI/bio11-src/ src and /JavaCard/JCBioAPI/bio11-src/doc_jc221 or /JavaCard/JCBioAPI/bio11-src/doc_jc222 for details).
- JAR files correspondent to the above-mentioned folders embodying class, export (EXP), Java source, etc. files (see the /JavaCard/JCBioAPI/jars for details). The JAR archive that carries org.javacardforum.javacard.biometry package compiled stub class files and export file (i.e. /JavaCard/JCBioAPI/jars/bio11.jar) should be included in the build path for compilation and conversion of the Java Card packages using the provided matcher library (e.g., for the applets from the /JavaCard/JCBioAPISamples directory).

The specific configuration full org.javacardforum.javacard.biometry implementation compiled binaries or natively integrated version of it are available for a wide range card projects. Please, contact Neurotechnology for details.

On-card biometric server (manager) and biometric client sample applets demonstrating the use of on-card matching library API are provided within /JavaCard/JCBioAPISamples/ directory containing:

- Applets class and export (EXP) files indispensable respectively for the compilation and further conversion of the Java Card packages that use the aforementioned applet packages.
- The converted applet (CAP) files of all sample applet packages that may be uploaded on a card having uploaded MegaMatcher On Card matching engine library in advance (see, e.g., /JavaCard/JCBioAPISamples/fingerSamples-cap/cap_jc221 or

/JavaCard/JCBioAPISamples/fingerSamples-cap/cap_jc222 for details).

- Additionally Java source files of the sample applets and full Javadoc of the fingerprint and / or face matcher library `org.javacardforum.javacard.biometry` API along with dependent sample applet packages are provided for evaluation and development purposes (see, e.g., /JavaCard/JCBioAPISamples/fingerSamples-src/src and /JavaCard/JCBioAPISamples/fingerSamples-src/doc_jc221 or /JavaCard/JCBioAPISamples/fingerSamples-src/doc_jc222 for details).
- JAR files correspondent to the above-mentioned folders embodying class, export (EXP), Java source, etc. files (see the /JavaCard/JCBioAPISamples/jars for details).

Notes

Please, follow the Javadoc from the /JavaCard/JCBioAPISamples/fingerSamples-src/doc_jc221 while development to follow support templates formats and restrictions upon pure Java level `org.javacardforum.javacard.biometry` library implementation usage. Also follow the CAP files loading and deletion order specified in one of the further sections. The concrete order of applets installation and packages deletion is also specified thereafter (see the section called JavaCard Part (see page 6) for details).

2.2 PC Part

The PC part of the MegaMatcher On Card 2.0 SDK consists of the following parts:

- MegaMatcher On Card PC-side Client Extractor (see page 152)
- Biometric Standards Support (see page 13) library for extracted templates conversion to the standard card format (see FMCard (see page 66))
- NSmartCards Library (see page 268) (Neurotec.SmartCards Library (see page 899) in .NET) for communication with a smartcard.
- Samples (see page 11) and tutorials (see page 10) showing how to perform fingerprint and / or face enrollment on and verification with a smartcard.

Notes

MegaMatcher On Card 2.0 SDK supports only standard FMCard (see page 66) formats.

3 Using

Since version 2 MegaMatcher On Card 2.0 SDK is a standalone SDK which includes main libraries (see API Reference (see page 13) for more information), programming samples and tutorials. Nevertheless, MegaMatcher On Card 2.0 SDK still can be used with MegaMatcher, VeriLook and VeriFinger SDKs. In this case VeriLook, MegaMatcher or VeriFinger SDK distribution files should be replaced by MegaMatcher On Card 2.0 SDK distribution files.

Notes

Do not replace NLICENSES.cfg file in MegaMatcher, VeriFinger or VeriLook SDK with the one from MegaMatcher On Card 2.0 SDK distribution. Otherwise, product will not retrieve a correct license.

3.1 Java Card Part

3.1.1 On-card Biometric APIs

Before the usage or further development of the on-card owner verification biometric API implementation you need to be familiar with the standards and specifications that are listed below.

Referenced International Standard Organization (ISO) standards

- [ISO/IEC 7816-3:2006. Identification cards -- Integrated circuit cards -- Part 3: Cards with contacts -- Electrical interface and transmission protocols.](#)
- [ISO/IEC 7816-4:2005. Identification cards -- Integrated circuit cards -- Part 4: Organization, security and commands for interchange.](#)
- [ISO/IEC 7816-6:2004. Identification cards -- Integrated circuit cards -- Part 6: Interindustry data elements for interchange.](#)
- [ISO/IEC 7816-9:2004. Identification cards -- Integrated circuit cards -- Part 9: Commands for card management.](#)
- [ISO/IEC 7816-11:2004. Identification cards -- Integrated circuit cards -- Part 11: Personal verification through biometric methods.](#)
- [ISO/IEC 19785-1:2006. Information technology -- Common Biometric Exchange Formats Framework -- Part 1: Data element specification.](#)

Referenced documents

- [Biometric Application Programming Interface \(API\) for Java Card v1.1.](#)
- [Runtime Environment Specification for the Java Card Platform v2.2.1.](#)
- [Virtual Machine Specification for the Java Card Platform v2.2.1.](#)
- [Application Programming Interface Specification for the Java Card Platform v2.2.1.](#)
- [Application Programming Interface Specification for the Java Card Platform v2.2.2.](#)
- [GlobalPlatform Card Specification v2.1.1.](#)

Advised Development Tools

- [J2SE SDK v1.4 / 1.5.](#)

- [Java Card Development Kit v2.2.1 / v2.2.2](#)
- [NXP JCOP tools plug-in v3.2.8 for Eclipse IDE](#).

The latter development tool provides functionality for working with any smart card implementing Java Card 2.1.1 /Java Card 2.2.2 and Open Platform 2.0.1 / Global Platform 2.1.1 specifications.

Please have a look at the documentation before using the biometric API implementation and samples provided (see /JavaCard/JCBioAPISamples/fingerSamples-src/ doc_jc221 for details).

3.1.1.1 Using On-Card Bio API Library

To use the implementation of on-card fingerprint and / or face templates matching library `org.javacardforum.javacard.biometry` while building proper Java Card projects, the stub class and export (EXP) files of the library must be reachable for the project for compilation and conversion purposes respectively. This may be achieved by including the /JavaCard/JCBioAPI/jars/bio11.jar in the build path of the developed project.

To make the library accessible on a Java Card or in the Java Card Runtime Environment (JCRE) emulator, the converted applet CAP file of the Bio API library (i.e. `biometry.cap`) must be uploaded in advance before any dependent CAP file of the project is attempted to load.

3.1.1.2 Using On-card Bio API Sample Applets

To use the provided biometric API implementation and samples on a real Java Card or in JCRE emulator, the converted applet (CAP) files of biometry library and / applet packages must be uploaded in the definite order (CAP files can be found in , e.g., the

/JavaCard/JCBioAPI/bio11-cap/cap_jc221 and

/JavaCard/ JCBioAPISamples/fingerSamples-cap/cap_jc221/ directories):

1. `biometry.cap` with AID = [0xA0:0x00:0x00:0x01:0x32:0x00:0x01] conforming to Biometric Application Programming Interface (API) for Java Card v 1.1;
2. `fingerServer.cap` with AID = [0xF0:0x4E:0x46:0x53:0x65:0x72:0x76:0x65:0x72] (*faceServer.cap for faces*);
3. `fingerClient.cap` with AID = [0xF0:0x4E:0x46:0x43:0x6C:0x69:0x65:0x6E:0x74] (*faceClient.cap for faces*).

Notes:

`biometry.cap` file is available for high volume projects only after personally contacting Neurotechnology.

Steps 2 and 3 may be replaced by loading merged sample applets package `fingerSample.cap` with AID = [0xF0:0x4E:0x46:0x53:0x61:0x6D:0x70:0x6C:0x65].

The order defined above is based on the chain of references from one package to the other, e.g., `com.neurotec.javacard.fingerServer` imports the `org.javacardforum.javacard.biometry` library package (see Runtime Environment Specification for the Java card Platform for details). The loading of CAP files correspondent to sample server and sample client packages may be reordered. Essential is the sequence of the server and client applets installation:

- 4.1. `com.neurotec.javacard.fingerServer.NFServerSample` with [0xF0:0x4E:0x46:0x53:0x65:0x72:0x76:0x65:0x72] (*com.neurotec.javacard.faceServer.NLServerSample for faces*).
- 4.2. `com.neurotec.javacard.fingerClient.NFClientSample` with AID = [0xF0:0x4E:0x46:0x43:0x6C:0x69:0x65:0x6E:0x74] (*com.neurotec.javacard.faceClient.NLClientSample for faces*).

Notes:

Steps 4.1 and 4.2 may be replaced by installing sample applets from package `fingerSample` package, i.e.

com.neurotec.javacard.fingerSample.NFSampleServer with AID =
 0xF0:0x4E:0x46:0x53:0x61:0x6D:0x70:0x6C:0x65:0x53:0x65:0x72:0x76:0x65:0x72]
 (com.neurotec.javacard.faceSample.NLSampleServer for faces) and com.neurotec.javacard.fingerSample.NFSampleClient
 with AID = [0xF0:0x4E:0x46:0x53:0x61:0x6D:0x70:0x6C:0x65:0x43:0x6C:0x69:0x65:0x6E:0x74]
 (com.neurotec.javacard.faceSample.NLSampleClient for faces) respectively.

Because the biometric client applet has to obtain a JCRE owned instance of the AID of the registered biometric server applet to obtain shareable services, the server applet must be installed first. Otherwise client will not be able to obtain shareable biometric matching functionality (see Runtime Environment Specification for the Java card Platform, section 6, the source codes and Javadoc of the com.neurotec.javacard.fingerServer and com.neurotec.javacard.fingerClient packages in JavaCard/JCBioAPISamples/fingerSamples-src/doc_jc221 directory for details). To test the functioning of the applets installed, the APDUs supported by the biometric server and biometric client applets have to be sent (see the sample biometric server and client applets process() method in the documentation in the Javadoc within aforementioned directory for details upon supported APDUs).

Pay attention that the deletion of the Java Card Bio API implementation library and sample applet packages has also to be performed in the concrete order (that is reverse to the installation consecution).

1. fingerClient.cap with AID = [0xF0:0x4E:0x46:0x43:0x6C:0x69:0x65:0x6E:0x74] (*faceClient.cap for faces matcher*);
2. fingerServer.cap with AID = [0xF0:0x4E:0x46:0x53:0x65:0x72:0x76:0x65:0x72] [14:45:37] (*faceServer.cap for faces matcher*);
3. biometry.cap with AID = [0xA0:0x00:0x00:0x01:0x32:0x00:0x01] .

Notes:

biometry.cap file is available for high volume projects only after personally contacting Neurotechnology.

The deletion order is based on the active references to the static memory fields or objects from the other packages (see Runtime Environment Specification for the Java card Platform, section 11 and documentation in /JavaCard/JCBioAPISamples/fingerSamples-src/doc_jc221/ or /JavaCard/JCBioAPISamples/faceSamples-src/ doc_jc221 directories for details).

Deletion of the sample server and client packages may be reordered, in case all references will be manually eliminated in advance by sending the ISO/IEC 7816-9:2004 DELETE FILE APDU command to the NFClientSample and NFServerSample (or *NLClientSample* and *NLServerSample for faces accordingly*) applets prior to deletion (see the sample biometric server and client applets process() method in the Javadoc placed in the aforementioned directories).

The installation and deletion of the packages on Java Card may be simplified by using SmartcardCapLoaderSample (see page 12) in case NXP JCOP tools plug-in Off-Card API libraries are available. In this case terminal sample may be applied for loading, installation, deletion of packages on-card or in the JCOP tools smartcard JCRE emulator (on Windows platform).

3.1.2 JC Bio API Matching On-Card Library Configurations

Different configurations of the JC Bio API v1.1. org.javacardforum.javacard.biometry library implementations are available including:

- different biometrics / multi-biometrics support:
 - faces:
 - pure Java level implementation for Java Cards (may be uploaded by an integrator as post-issuance library);
 - customization for pure native OS masked solutions on demand.
 - fingers:

- different integration level:
 - pure Java level implementation for Java Cards (may be uploaded by an integrator as post-issuance library)
 - native level (masked into ROM) implementation for Java Cards;
 - customization for pure native OS masked solutions on demand;
- different speed modes;
- different persistent and transient memory consumption requirements.
- different accuracy modes;
 - fingers and faces (see the configurations options above).

Please contact Neurotechnology for the samples and optimal `org.javacardforum.javacard.biometry` library solution depending on the own top priorities within configuration options listed above.

The current sample biometric server and client finger and / or face applets from `/JavaCard/JCBioAPISamples` directory support the enrollment and verification of up to 2 fingerprint templates and single face template respectively. User can modify or create its own biometric server and client applications that support more / less templates management depending on the intensions and persistent memory resources.

3.2 PC Part

See Samples (see page 11) section on how to use the MegaMatcher On Card 2.0 SDK.

4 Tutorials

MegaMatcher On Card 2.0 SDK includes tutorials which can be used to explore the main functionality of product.

These tutorials are available:

Match on Smartcard

Tutorial	Description
SmartcardFingersEnroll	Demonstrates fingerprint enrollment to smartcard.
SmartcardFingersVerify	Demonstrates fingerprint verification on smartcard.
SmartcardFingersMultiEnroll	Standalone tutorial demonstrating multiple extracted standard fingerprint templates enrollment to smartcard.
SmartcardFingersMultiVerify	Standalone tutorial demonstrating multiple extracted standard fingerprint templates verification on a smartcard.
SmartcardFacesVerify	Demonstrates faces verification on smartcard.
SmartcardFacesEnroll	Demonstrates faces enrollment to smartcard.

Tutorials source files are saved in \Tutorials directory of SDK. For building C tutorials Microsoft Visual Studio is required (solution files are included).

For building and running Java tutorial the following tools are required (see instructions how to build Java tutorials within ReadMe.txt in Tutorials/MatchOnSmartcard/Java directory):

- Apache Ant
- At least one of the following utilities:
 - Java SDK 1.6;
 - Java SDK 1.5 / 1.6 and NXP JCOP tools v3.2.8 plug-in for Eclipse IDE;
 - Java SDK 1.5 / 1.6 and Java Card Development Kit v2.2.2.

For running JCDK tutorials scripts Java Card Development Kit v2.2.2 is required. The JCDK tutorials are supported on Windows platform only.

For running JCOP tutorials scripts NXP JCOP tools v3.2.8 plug-in for Eclipse IDE is required.

5 Samples

MegaMatcher On Card 2.0 SDK includes these sample programs which demonstrate how to use the product:

- Smartcard Fingers Client (see page 11) - C# and VB.NET sample which demonstrates how to enroll fingerprints to a smartcard and perform verification.
- Smartcard Faces Client (see page 11) - C# and VB.NET sample which demonstrates how to enroll facial features to a smartcard and perform verification.
- Smartcard CAP Loader (see page 12) - Java sample program used to upload specified Java Card converted applet (CAP) packages and install their application on a smartcard (or simulation).

5.1 Smartcard Fingers Client

Sample is intended to show how to enroll fingerprint(s) to a smartcard and perform verification of templates on a smartcard.

Files

Binaries of Smartcard Client sample (for C# and VB.NET programming language) are located:

- \Bin\Win32_x86\SmartcardFingersClientCS.exe
- \Bin\Win32_x86\SmartcardFingersClientVB.exe

Source are located:

- \Samples\MatchOnSmartcard\CS\SmartcardFingersClientCS
- \Samples\MatchOnSmartcard\VB\SmartcardFingersClientVB

5.2 Smartcard Faces Client

Sample is intended to show how to enroll face features to a smartcard and perform verification of templates on a smartcard.

Files

Binaries of Smartcard Client sample are located:

- Bin\Win32_x86\SmartcardFacesClientCS.exe
- Bin\Win32_x86\SmartcardFacesClientVB.exe

Source are located:

- Samples\MatchOnSmartcard\CS\SmartcardFacesClientCS\
- Samples\MatchOnSmartcard\VB\SmartcardFacesClientVB\

5.3 Smartcard CAP Loader

This sample application demonstrates how to use the NXP JCOP (Java Card Operating Platform) tools plug-in v 3.2.8 for Eclipse IDE Off-card API libraries to provide functionality for communication with a Global Platform 2.1.1 /Open Platform 2.0.1 specifications compatible Java Cards. The sample provides the following features:

- get information about the current Java Card status
- download a CAP-file on a Java Card and install all related applets
- delete package and all related objects from a Java Card.

Requirements for building and running *Smartcard CAP Loader* sample application:

- Java 2 SDK version:
 - 1.4/1.5 for older NXP JCOP tools versions (e.g., v3.1.2)
 - 1.5/1.6. for recent NXP JCOP tools versions (e.g., v3.2.8)
- JCOP tools plug-in for Eclipse IDE off-card API Java and native libraries:
 - offcard.jar
 - crypto.jar (required in recent NXP JCOP tools versions (e.g., v3.2.8), but no needed in older ones (e.g., v3.1.2)
 - jpcsc.jar (it is present within older JCOP tools plug-in version or may be obtained from <http://www.linuxnet.com/middleware/files/jpcsc-0.8.0-src.zip>)
 - libjpcsc.so file for Linux or jpcsc.dll for Windows (it is present within older JCOP tools plug-in version corresponding OS folder or may be obtained from <http://www.linuxnet.com/middleware/files/jpcsc-0.8.0-src.zip>).

Files

Binaries of Smartcard CAP Loader sample (for Java programming language) are located:

- \Bin\Win32_x86\SmartcardCapLoaderSample.jar

Source are located:

- \Samples\MatchOnSmartcard\Java\SmartcardCapLoaderSample

Notes

\doc folder contains index.html file which provides detailed information how to use this sample.

6 API Reference

Modules

Name	Description
.NET Reference (see page 606)	This chapter contains reference of all libraries included in MegaMatcher On Card 2.0 SDK for .NET developers. C# language is used where it is needed to provide sample code.

6.1 C/C++ Reference

This chapter contains reference of all libraries included in MegaMatcher On Card 2.0 SDK for developers using C/C++ language.

6.1.1 NBiometricStandards Library

Provides information and API Reference of supported biometric standards by MegaMatcher On Card 2.0 SDK.

Remarks

Requirements (Windows)

- **Import library:** NBiometricStandards.dll.lib
- **DLL:** NBiometricStandards.dll

Requirements (Linux)

- **Shared object:** libNBiometricStandards.so

Notes

NBiometricStandards library for MegaMatcher On Card 2.0 SDK supports only Neurotechnology proprietary to/from FMCard templates conversion.

Modules

Name	Description
Bdif Module (see page 14)	Provides library registration information.
BdifTypes Module (see page 14)	Contains types and enumerations for biometric data interchange.
CbeffBdbFormatIdentifiers Module (see page 37)	Specifies CBEFF Biometric Data Block (BDB) format identifiers registered with International Biometric Industry Association (IBIA).
CbeffBiometricOrganizations Module (see page 48)	Specifies CBEFF biometric organizations identifiers registered with International Biometric Industry Association (IBIA).
FMCard Module (see page 66)	Provides functionality for converting ISO/IEC 19794-2:2005 card format standard templates to and from Neurotechnology Finger Records (NFRRecords).
NBiometricStandards Module (see page 73)	Provides information about the NBiometricsStandards library.

6.1.1.1 Bdif Module

Provides library registration information.

Files

Name	Description
Bdif.h (see page 14)	Header file for Bdif module. Provides library registration information.

6.1.1.1.1 Files

6.1.1.1.1.1 Bdif.h

Header file for Bdif module. Provides library registration information.

Module

Bdif Module ([see page 14](#))

6.1.1.2 BdifTypes Module

Contains types and enumerations for biometric data interchange.

Files

Name	Description
BdifTypes.h (see page 36)	Header file for BdifTypes. Contains types and enumerations.

Macros

Name	Description
BDIF_DO_NOT_CHECK_CBEFF_PRODUCT_ID (see page 35)	The flag which is used to indicate not check Cbeff product ID.

Structs, Records, Enums

	Name	Description
	BdifEyeColor_ (see page 15)	Enumerates eye color constants.
	BdifEyePosition_ (see page 17)	Enumerates eye position constants.
	BdifFaceExpression_ (see page 17)	Enumerates facial feature constants.
	BdifFaceFeaturePoint_ (see page 18)	Enumerates face feature point constants.
	BdifFaceFeaturePointType_ (see page 19)	Enumerates face feature point type constants.
	BdifFaceProperties_ (see page 19)	Enumerates basic face properties constants.
	BdifFPatternClass_ (see page 21)	Enumerates fingerprint pattern class constants.
	BdifFPImpressionType_ (see page 22)	Enumerates fingerprint impression type constants.
	BdifFPMinutiaNeighbor_ (see page 25)	Enumerates fingerprint minutia type neighbors.
	BdifFPMinutiaType_ (see page 25)	Enumerates fingerprint minutia type constants.
	BdifFPPosition_ (see page 26)	Enumerates fingerprint position constants.

	BdifGender_ (see page 28)	Enumerates gender constants.
	BdifHairColor_ (see page 29)	Enumerates hair color constants.
	BdifImageSourceType_ (see page 31)	Enumerates image source type constants.
	BdifIrisOrientation_ (see page 32)	Enumerates iris orientation constants.
	BdifIrisScanType_ (see page 33)	Enumerates iris scan type constants.
	BdifScaleUnits_ (see page 34)	Enumerates scale unit constants.
	BdifStandard_ (see page 35)	Enumerates standard constants.
	BdifEyeColor (see page 15)	Enumerates eye color constants.
	BdifEyePosition (see page 17)	Enumerates eye position constants.
	BdifFaceExpression (see page 17)	Enumerates facial feature constants.
	BdifFaceFeaturePoint (see page 18)	Enumerates face feature point constants.
	BdifFaceFeaturePointType (see page 19)	Enumerates face feature point type constants.
	BdifFaceProperties (see page 19)	Enumerates basic face properties constants.
	BdifFPatternClass (see page 21)	Enumerates fingerprint pattern class constants.
	BdifFPImpressionType (see page 22)	Enumerates fingerprint impression type constants.
	BdifFPMinutiaNeighbor (see page 25)	Enumerates fingerprint minutia type neighbors.
	BdifFPMinutiaType (see page 25)	Enumerates fingerprint minutia type constants.
	BdifFPPosition (see page 26)	Enumerates fingerprint position constants.
	BdifGender (see page 28)	Enumerates gender constants.
	BdifHairColor (see page 29)	Enumerates hair color constants.
	BdifImageSourceType (see page 31)	Enumerates image source type constants.
	BdifIrisOrientation (see page 32)	Enumerates iris orientation constants.
	BdifIrisScanType (see page 33)	Enumerates iris scan type constants.
	BdifScaleUnits (see page 34)	Enumerates scale unit constants.
	BdifStandard (see page 35)	Enumerates standard constants.

6.1.1.2.1 Structs, Records, Enums

6.1.1.2.1.1 BdifEyeColor Enumeration

Enumerates eye color constants.

C++

```
typedef enum BdifEyeColor_ {
    becUnspecified = 0,
    becBlack = 1,
    becBlue = 2,
    becBrown = 3,
    becGray = 4,
    becGreen = 5,
    becHazel = 6,
    becMaroon = 7,
    becMultiColored = 8,
    becPink = 9,
    becUnknown = 255
} BdifEyeColor;
```


Members

Members	Description
becUnspecified = 0	Unspecified.
becBlack = 1	Black.
becBlue = 2	Blue.
becBrown = 3	Brown.
becGray = 4	Gray.
becGreen = 5	Green.
becHazel = 6	Hazel.
becMaroon = 7	Maroon.
becMultiColored = 8	Multi colored.
becPink = 9	Pink.
becUnknown = 255	Unknown.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
becUnspecified	-	+	-	-	-	+	-	-	+	-
becBlack	-	+	-	-	-	-	-	-	+	-
becBlue	-	+	-	-	-	+	-	-	+	-
becBrown	-	+	-	-	-	+	-	-	+	-
becGray	-	+	-	-	-	-	-	-	+	-
becGreen	-	+	-	-	-	+	-	-	+	-
becHazel	-	+	-	-	-	+	-	-	-	-
becMaroon	-	+	-	-	-	+	-	-	-	-
becMulti-Colored	-	+	-	-	-	+	-	-	+	-
becPink	-	+	-	-	-	+	-	-	+	-
becUnknown	-	+	-	-	-	+	-	-	+	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.2 BdifEyePosition Enumeration

Enumerates eye position constants.

C++

```
typedef enum BdifEyePosition_ {
    bepUnknown = 0,
    bepRight = 1,
    bepLeft = 2
} BdifEyePosition;
```

Members

Members	Description
bepUnknown = 0	Unknown eye position.
bepRight = 1	Right eye.
bepLeft = 2	Left eye.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bepUnknown	-	+	-	+	-	-	-	-	-	+
bepRight	-	+	-	+	-	-	-	-	-	+
bepLeft	-	+	-	+	-	-	-	-	-	+

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module ([see page 14](#))

6.1.1.2.1.3 BdifFaceExpression Enumeration

Enumerates facial feature constants.

C++

```
typedef enum BdifFaceExpression_ {
    bfeUnspecified = 0,
    bfeNeutral = 1,
    bfeSmile = 2,
    bfeSmileOpenedJaw = 3,
    bfeRaisedBrows = 4,
    bfeEyesAway = 5,
    bfeSquinting = 6,
```

```
    bfeFrowning = 7;
} BdifFaceExpression;
```

Members

Members	Description
bfeUnspecified = 0	Unspecified.
bfeNeutral = 1	Neutral.
bfeSmile = 2	Smile.
bfeSmileOpenedJaw = 3	Smile with jaw open.
bfeRaisedBrows = 4	Eyebrows risen.
bfeEyesAway = 5	Eyes looking away.
bfeSquinting = 6	Squinting.
bfeFrowning = 7	Frowning.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bfeUnspecified	-	+	-	-	-	+	-	-	+	-
bfeNeutral	-	+	-	-	-	+	-	-	+	-
bfeSmile	-	+	-	-	-	+	-	-	+	-
bfeSmileOpenedJaw	-	-	-	-	-	+	-	-	+	-
bfeRaisedBrows	-	+	-	-	-	+	-	-	+	-
bfeEyesAway	-	+	-	-	-	+	-	-	+	-
bfeSquinting	-	+	-	-	-	+	-	-	+	-
bfeFrowning	-	+	-	-	-	+	-	-	+	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.4 BdifFaceFeaturePoint Structure

Enumerates face feature point constants.

C++

```
typedef struct BdifFaceFeaturePoint_ {
    BdifFaceFeaturePointType Type;
```

```
NByte Code;  
NUShort X;  
NUShort Y;  
} BdifFaceFeaturePoint;
```

Members

Members	Description
BdifFaceFeaturePointType Type;	Face feature point type.
NByte Code;	Face feature point code.
NUShort X;	X value of feature point value.
NUShort Y;	Y value of face feature point.

Module

BdifTypes Module (see page 14)

6.1.1.2.1.5 BdifFaceFeaturePointType Enumeration

Enumerates face feature point type constants.

C++

```
typedef enum BdifFaceFeaturePointType_ {  
    bffptPoint2D = 1  
} BdifFaceFeaturePointType;
```

Members

Members	Description
bffptPoint2D = 1	2D point.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bffptPoint2D	-	+	-	-	-	+	-	-	+	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.6 BdifFaceProperties Enumeration

Enumerates basic face properties constants.

C++

```
typedef enum BdifFaceProperties_ {
    bfpNotSpecified = 0,
    bfpSpecified = 0x000001,
    bfpGlasses = 0x000002,
    bfpMoustache = 0x000004,
    bfpBeard = 0x000008,
    bfpTeethVisible = 0x000010,
    bfpBlink = 0x000020,
    bfpMouthOpen = 0x000040,
    bfpLeftEyePatch = 0x000080,
    bfpRightEyePatch = 0x000100,
    bfpBothEyePatch = 0x000200,
    bfpDarkGlasses = 0x000400,
    bfpDistortingCondition = 0x000800,
    bfpHat = 0x01000000,
    bfpScarf = 0x02000000,
    bfpNoEar = 0x04000000
} BdifFaceProperties;
```

Members

Members	Description
bfpNotSpecified = 0	Face properties not specified.
bfpSpecified = 0x000001	Face properties specified.
bfpGlasses = 0x000002	Wearing glasses.
bfpMoustache = 0x000004	Moustaches.
bfpBeard = 0x000008	Beard.
bfpTeethVisible = 0x000010	Teeth visible.
bfpBlink = 0x000020	Eye blink.
bfpMouthOpen = 0x000040	Mouth is open.
bfpLeftEyePatch = 0x000080	Left eye patch.
bfpRightEyePatch = 0x000100	Right eye patch.
bfpBothEyePatch = 0x000200	Both eyes patch.
bfpDarkGlasses = 0x000400	Wearing dark glasses.
bfpDistortingCondition = 0x000800	Distorting conditions.
bfpHat = 0x01000000	Wearing a hat.
bfpScarf = 0x02000000	Wearing a scarf.
bfpNoEar = 0x04000000	No ear.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bfpNotSpecified	+	+	-	-	-	+	-	-	+	-
bfpSpecified	-	-	-	-	-	+	-	-	+	-
bfpGlasses	+	+	-	-	-	+	-	-	+	-
bfpMoustache	-	+	-	-	-	+	-	-	+	-
bfpBeard	-	+	-	-	-	+	-	-	+	-
bfpTeethVisible	-	+	-	-	-	+	-	-	+	-
bfpBlink	-	+	-	-	-	+	-	-	+	-
bfpMouthOpen	-	+	-	-	-	+	-	-	+	-
bfpLeftEyePatch	-	+	-	-	-	+	-	-	+	-

bfpBothEyePatch	-	-	-	-	-	+	-	-	-	-
bfpDarkGlasses	-	+	-	-	-	+	-	-	+	-
bfpDistortingCondition	-	+	-	-	-	+	-	-	+	-
bfpHat	+	+	-	-	-	-	-	-	-	-
bfpScarf	+	+	-	-	-	-	-		-	-
bfpNoEar		+	-	-	-	-	-			-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.7 BdifFPatternClass Enumeration

Enumerates fingerprint pattern class constants.

C++

```
typedef enum BdifFPatternClass_ {
    bfpcUnknown = 0,
    bfpcPlainArch = 1,
    bfpcTentedArch = 2,
    bfpcRadialLoop = 3,
    bfpcUlnarLoop = 4,
    bfpcPlainWhorl = 5,
    bfpcCentralPocketLoop = 6,
    bfpcDoubleLoop = 7,
    bfpcAccidentalWhorl = 8,
    bfpcWhorl = 9,
    bfpcRightSlantLoop = 10,
    bfpcLeftSlantLoop = 11,
    bfpcScar = 12,
    bfpcAmputation = 15
} BdifFPatternClass;
```

Members

Members	Description
bfpcUnknown = 0	Unknown.
bfpcPlainArch = 1	Plain arch.
bfpcTentedArch = 2	Tented arch.
bfpcRadialLoop = 3	Radial loop.
bfpcUlnarLoop = 4	Ulnar loop.
bfpcPlainWhorl = 5	Plain whorl.

bfpcCentralPocketLoop = 6	Central pocker loop.
bfpcDoubleLoop = 7	Double loop.
bfpcAccidentalWhorl = 8	Accidental Whorl.
bfpcWhorl = 9	Whorl.
bfpcRightSlantLoop = 10	Right Slant Loop.
bfpcLeftSlantLoop = 11	Left slant loop.
bfpcScar = 12	Scar.
bfpcAmputation = 15	Amputation.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bfpcUnknown	+	+	-	-	-	-	-	-	-	-
bfpcPlainArch	+	+	-	-	-	-	-	-	-	-
bfpcTentedArch	+	+	-	-	-	-	-	-	-	-
bfpcRadialLoop	+	+	-	-	-	-	-	-	-	-
bfpcUlnarLoop	+	+	-	-	-	-	-	-	-	-
bfpcPlainWhorl	+	+	-	-	-	-	-	-	-	-
bfpcCentralPocketLoop	+	+	-	-	-	-	-	-	-	-
bfpcDoubleLoop	+	+	-	-	-	-	-	-	-	-
bfpcAccidentalWhorl	+	+	-	-	-	-	-	-	-	-
bfpcWhorl	+	+	-	-	-	-	-	-	-	-
bfpcRightSlantLoop	+	+	-	-	-	-	-	-	-	-
bfpcLeftSlantLoop	+	+	-	-	-	-	-	-	-	-
bfpcScar	+	+	-	-	-	-	-	-	-	-
bfpcAmputation	+	+	-	-	-	-	-	-	-	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module ([see page 14](#))

6.1.1.2.1.8 BdifFPImpressionType Enumeration

Enumerates fingerprint impression type constants.

C++

```
typedef enum BdifFPImpressionType_ {
    bfpitLiveScanPlain = 0,
    bfpitLiveScanRolled = 1,
    bfpitNonliveScanPlain = 2,
    bfpitNonliveScanRolled = 3,
    bfpitLatentImpression = 4,
    bfpitLatentTracing = 5,
    bfpitLatentPhoto = 6,
    bfpitLatentLift = 7,
    bfpitLiveScanVerticalSwipe = 8,
    bfpitLiveScanContactless = 9,
    bfpitLiveScanPalm = 10,
    bfpitNonliveScanPalm = 11,
    bfpitLatentPalmImpression = 12,
    bfpitLatentPalmTracing = 13,
    bfpitLatentPalmPhoto = 14,
    bfpitLatentPalmLift = 15,
    bfpitLiveScanOpticalContactPlain = 20,
    bfpitLiveScanOpticalContactRolled = 21,
    bfpitLiveScanNonOpticalContactPlain = 22,
    bfpitLiveScanNonOpticalContactRolled = 23,
    bfpitLiveScanOpticalContactlessPlain = 24,
    bfpitLiveScanOpticalContactlessRolled = 25,
    bfpitLiveScanNonOpticalContactlessPlain = 26,
    bfpitLiveScanNonOpticalContactlessRolled = 27,
    bfpitOther = 28,
    bfpitUnknown = 29
} BdifFPImpressionType;
```

Members

Members	Description
bfpitLiveScanPlain = 0	Plain live scan.
bfpitLiveScanRolled = 1	Rolled live scan.
bfpitNonliveScanPlain = 2	Plain non live scan.
bfpitNonliveScanRolled = 3	Rolled non live scan.
bfpitLatentImpression = 4	Latent impression.
bfpitLatentTracing = 5	Latent tracing.
bfpitLatentPhoto = 6	Latent photo.
bfpitLatentLift = 7	Latent lift.
bfpitLiveScanVerticalSwipe = 8	Vertical swipe live scan.
bfpitLiveScanContactless = 9	Contactless live scan.
bfpitLiveScanPalm = 10	Palm live scan.
bfpitNonliveScanPalm = 11	Palm non live scan.
bfpitLatentPalmImpression = 12	Latent palm impression.
bfpitLatentPalmTracing = 13	Latent palm tracing.
bfpitLatentPalmPhoto = 14	Latent palm photo.
bfpitLatentPalmLift = 15	Latent palm lift.
bfpitLiveScanOpticalContactPlain = 20	Non optical contact plain live scan.
bfpitLiveScanOpticalContactRolled = 21	Optical contact rolled live scan.
bfpitLiveScanNonOpticalContactPlain = 22	Non optical contactless plain live scan.
bfpitLiveScanNonOpticalContactRolled = 23	Non optical contact rolled live scan.
bfpitLiveScanOpticalContactlessPlain = 24	Optical contactless plain live scan.
bfpitLiveScanOpticalContactlessRolled = 25	Optical contactless rolled live scan.
bfpitLiveScanNonOpticalContactlessPlain = 26	Non optical contact plain live scan.
bfpitLiveScanNonOpticalContactlessRolled = 27	Non optical contact rolled live scan.

bfpitOther = 28	Other.
bfpitUnknown = 29	Unknown.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bfpitLiveScanPlain	+	+	+	-	+	-	+	+		-
bfpitLiveScanRolled	+	+	+	-	+	-	+	+		-
bfpitNonliveScanPlain	+	+	+	-	+	-	+	+		-
bfpitNonliveScanRolled	+	+	+	-	+	-	+	+		-
bfpitLatentImpression	+	+	-	-	+	-	-			-
bfpitLatentTracing	+	+	-	-	-	-	-		-	-
bfpitLatentPhoto	+	+	-	-	-	-	-	-	-	-
bfpitLatentLift	+	+	-	-	+	-	-	+	-	-
bfpitLiveScanVerticalSwipe	-	+	+	-	+	-	+	+	-	-
bfpitLiveScanContactless	-	-	+	-	+	-	-	+	-	-
bfpitLiveScanPalm	+	+	-	-	-	-	-	-	-	-
bfpitNonliveScanPalm	+	+	-	-	-	-	-	-	-	-
bfpitLatentPalmImpression	+	+	-	-	-	-	-	-	-	-
bfpitLatentPalmTracing	+	+	-	-	-	-	-	-	-	-
bfpitLatentPalmPhoto	+	+	-	-	-	-	-	-	-	-
bfpitLatentPalmLift	+	+	-	-	-	-	-	-	-	-
bfpitLiveScanOpticalContactPlain	-	+	-	-	-	-	-	-	-	-
bfpitLiveScanOpticalContactRolled	-	+	-	-	-	-	-	-	-	-
bfpitLiveScanNonOpticalContactPlain	-	+	-	-	-	-	-	-	-	-
bfpitLiveScanNonOpticalContactRolled	-	+	-	-	-	-	-	-	-	-
bfpitLiveScanOpticalContactlessPlain	-	+	-	-	-	-	-	-	-	-
bfpitLiveScanOpticalContactlessRolled	-	+	-	-	-	-	-	-	-	-
bfpitLiveScanNonOpticalContactlessPlain	-	+	-	-	-	-	-	-	-	-
bfpitLiveScanNonOpticalContactlessRolled	-	+	-	-	-	-	-	-	-	-
bfpitOther	-	+	-	-	-	-	-	-	-	-
bfpitUnknown	-	+	-	-	-	-	-	-	-	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8

- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.9 BdifFPMinutiaNeighbor Structure

Enumerates fingerprint minutia type neighbors.

C++

```
typedef struct BdifFPMinutiaNeighbor_ {
    NInt Index;
    NByte RidgeCount;
} BdifFPMinutiaNeighbor;
```

Members

Members	Description
NInt Index;	Index.
NByte RidgeCount;	Ridge count.

Module

BdifTypes Module (see page 14)

6.1.1.2.1.10 BdifFPMinutiaType Enumeration

Enumerates fingerprint minutia type constants.

C++

```
typedef enum BdifFPMinutiaType_ {
    bfpmtUnspecified = -1,
    bfpmtUnknown = 0,
    bfpmtEnd = 1,
    bfpmtBifurcation = 2,
    bfpmtOther = 3
} BdifFPMinutiaType;
```

Members

Members	Description
bfpmtUnspecified = -1	Unspecified minutia type.
bfpmtUnknown = 0	Unknown minutia type.
bfpmtEnd = 1	Ridge end minutia type.
bfpmtBifurcation = 2	Ridge bifurcation minutia type.
bfpmtOther = 3	bfpmtOther

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bfpmtUnspecified	+	+	-	-	-	-	-	-	-	-
bfpmtUnknown	+	+	-	-	-	-	-	-	-	-
bfpmtEnd	+	+	+	-	-	-	+	-	-	-
bfpmtBifurcation	+	+	+	-	-	-	+	-	-	-
bfpmtOther	+	+	+	-	-	-	+	-	-	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.11 BdifFPPosition Enumeration

Enumerates fingerprint position constants.

C++

```
typedef enum BdifFPPosition_ {
    bfppUnknown = 0,
    bfppRightThumb = 1,
    bfppRightIndexFinger = 2,
    bfppRightMiddleFinger = 3,
    bfppRightRingFinger = 4,
    bfppRightLittleFinger = 5,
    bfppLeftThumb = 6,
    bfppLeftIndexFinger = 7,
    bfppLeftMiddleFinger = 8,
    bfppLeftRingFinger = 9,
    bfppLeftLittleFinger = 10,
    bfppPlainRightThumb = 11,
    bfppPlainLeftThumb = 12,
    bfppPlainRightFourFingers = 13,
    bfppPlainLeftFourFingers = 14,
    bfppPlainThumbs = 15,
    bfppMajorCase = 19,
    bfppUnknownPalm = 20,
    bfppRightFullPalm = 21,
    bfppRightWritersPalm = 22,
    bfppLeftFullPalm = 23,
    bfppLeftWritersPalm = 24,
    bfppRightLowerPalm = 25,
    bfppRightUpperPalm = 26,
    bfppLeftLowerPalm = 27,
    bfppLeftUpperPalm = 28,
    bfppRightOther = 29,
    bfppLeftOther = 30,
    bfppRightInterdigital = 31,
    bfppRightThenar = 32,
    bfppRightHypothenar = 33,
    bfppLeftInterdigital = 34,
    bfppLeftThenar = 35,
    bfppLeftHypothenar = 36
} BdifFPPosition;
```

Members

Members	Description
bfppUnknown = 0	Unknown.
bfppRightThumb = 1	Right thumb.
bfppRightIndexFinger = 2	Right index finger.
bfppRightMiddleFinger = 3	Right middle finger.
bfppRightRingFinger = 4	Right ring finger.
bfppRightLittleFinger = 5	Right little finger.
bfppLeftThumb = 6	Left thumb.
bfppLeftIndexFinger = 7	Left index finger.
bfppLeftMiddleFinger = 8	Left middle finger.
bfppLeftRingFinger = 9	Left ring finger.
bfppLeftLittleFinger = 10	Left little finger.
bfppPlainRightThumb = 11	Plain right thumb.
bfppPlainLeftThumb = 12	Plain left thumb.
bfppPlainRightFourFingers = 13	Plain right four fingers.
bfppPlainLeftFourFingers = 14	Plain left four fingers.
bfppPlainThumbs = 15	Plain thumbs.
bfppMajorCase = 19	Major case.
bfppUnknownPalm = 20	Unknown palm.
bfppRightFullPalm = 21	Right full palm.
bfppRightWritersPalm = 22	Right writers palm.
bfppLeftFullPalm = 23	Left full palm.
bfppLeftWritersPalm = 24	Left writers palm.
bfppRightLowerPalm = 25	Right little palm.
bfppRightUpperPalm = 26	Right upper palm.
bfppLeftLowerPalm = 27	Left lower palm.
bfppLeftUpperPalm = 28	Left upper palm.
bfppRightOther = 29	Right other.
bfppLeftOther = 30	Left other.
bfppRightInterdigital = 31	Right interdigital.
bfppRightThenar = 32	Right thenar.
bfppRightHypothenar = 33	Right hypothenar.
bfppLeftInterdigital = 34	Left interdigital.
bfppLeftThenar = 35	Left thenar.
bfppLeftHypothenar = 36	Right hypothenar.

Remarks

Members compatibility with various formats:

Member	1	2	3	4	5	6	7	8	9	10
bfppUnknown	+	+	+	-	+	-	+	+	-	-
bfppRightThumb	+	+	+	-	+	-	+	+	-	-
bfppLeftIndexFinger	+	+	+	-	+	-	+	+	-	-
bfppLeftMiddleFinger	+	+	+	-	+	-	+	+	-	-
bfppLeftRingFinger	+	+	+	-	+	-	+	+	-	-
bfppLeftLittleFinger	+	+	+	-	+	-	+	+	-	-

bfppPlainRightThumb	+	+	+	-	-	-	-	-	-	-
bfppPlainLeftThumb	+	+	-	-	-	-	-	-	-	-
bfppPlainRightFourFingers	+	+	-	-	+	-	-	+	-	-
bfppPlainLeftFourFingers	+	+	-	-	+	-	-	+	-	-
bfppPlainThumbs	-	+	-	-	+	-	-	+	-	-
bfppMajorCase	-	+	-	-	-	-	-	-	-	-
bfppUnknownPalm	+	+	-	-	+	-	-	+	-	-
bfppRightFullPalm	+	+	-	-	+	-	-	+	-	-
bfppRightWritersPalm	+	+	-	-	+	-	-	+	-	-
bfppLeftFullPalm	+	+	-	-	+	-	-	+	-	-
bfppLeftWritersPalm	+	+	-	-	+	-	-	+	-	-
bfppRightLowerPalm	+	+	-	-	+	-	-	+	-	-
bfppRightUpperPalm	+	+	-	-	+	-	-	+	-	-
bfppLeftLowerPalm	+	+	-	-	+	-	-	+	-	-
bfppLeftUpperPalm	+	+	-	-	+	-	-	+	-	-
bfppRightOther	+	+	-	-	+	-	-	+	-	-
bfppLeftOther	+	+	-	-	+	-	-	+	-	-
bfppRightInterdigital	-	+	-	-	+	-	-	+	-	-
bfppRightThenar	-	+	-	-	+	-	-	+	-	-
bfppRightHypothenar	-	+	-	-	+	-	-	+	-	-
bfppLeftInterdigital	-	+	-	-	+	-	-	+	-	-
bfppLeftThenar	-	+	-	-	+	-	-	+	-	-
bfppLeftHypothenar	-	+	-	-	+	-	-	+	-	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.12 BdifGender Enumeration

Enumerates gender constants.

C++

```
typedef enum BdifGender_ {
    bgUnspecified = 0,
    bgMale = 1,
    bgFemale = 2,
    bgUnknown = 255
} BdifGender;
```

Members

Members	Description
bgUnspecified = 0	Unspecified.
bgMale = 1	Male
bgFemale = 2	Female.
bgUnknown = 255	Unknown.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bgUnspecified	-	-	-	-	-	+	-	-	+	-
bgMale	-	-	-	-	-	+	-	-	+	-
bgFemale	-	-	-	-	-	+	-	-	+	-
bgUnknown	-	-	-	-	-	+	-	-	+	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.13 BdifHairColor Enumeration

Enumerates hair color constants.

C++

```
typedef enum BdifHairColor_ {
    bhcUnspecified = 0,
    bhcBald = 1,
    bhcBlack = 2,
    bhcBlonde = 3,
    bhcBrown = 4,
    bhcGray = 5,
    bhcRed = 6,
}
```

```

bhcBlue = 7,
bhcGreen = 8,
bhcOrange = 9,
bhcPink = 10,
bhcPurple = 11,
bhcSandy = 12,
bhcAuburn = 13,
bhcWhite = 14,
bhcStrawberry = 15,
bhcUnknown = 255
} BdifHairColor;

```

Members

Members	Description
bhcUnspecified = 0	Unspecified.
bhcBald = 1	Bald.
bhcBlack = 2	Black.
bhcBlonde = 3	Blonde.
bhcBrown = 4	Brown.
bhcGray = 5	Gray.
bhcRed = 6	Red.
bhcBlue = 7	Blue.
bhcGreen = 8	Green.
bhcOrange = 9	Orange.
bhcPink = 10	Pink.
bhcPurple = 11	Purple.
bhcSandy = 12	Sandy.
bhcAuburn = 13	Auburn.
bhcWhite = 14	White.
bhcStrawberry = 15	Strawberry.
bhcUnknown = 255	Unknown.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bhcUnspecified	-	+	-	-	-	+	-	-	+	-
bhcBald	-	+	-	-	-	+	-	-	+	-
bhcBlack	-	+	-	-	-	+	-	-	+	-
bhcBlonde	-	+	-	-	-	+	-	-	+	-
bhcBrown	-	+	-	-	-	+	-	-	+	-
bhcGray	-	+	-	-	-	+	-	-	+	-
bhcRed	-	+	-	-	-	+	-	-	+	-
bhcBlue	-	+	-	-	-	+	-	-	-	-
bhcGreen	-	+	-	-	-	+	-	-	-	-
bhcOrange	-	+	-	-	-	+	-	-	-	-
bhcPink	-	+	-	-	-	+	-	-	-	-
bhcPurple	-	+	-	-	-	-	-	-	-	-
bhcSandy	-	+	-	-	-	+	-	-	-	-
bhcAuburn	-	+	-	-	-	+	-	-	-	-

bhcWhite	-	+	-	-	-	+	-	-	+	-
bhcStrawberry	-	+	-	-	-	+	-	-	-	-
bhcUnknown	-	-	-	-	-	-	-	-	-	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.14 BdifImageSourceType Enumeration

Enumerates image source type constants.

C++

```
typedef enum BdifImageSourceType_ {
    bistUnspecified = 0,
    bistUnknownPhoto = 1,
    bistDigitalPhoto = 2,
    bistScannedPhoto = 3,
    bistUnknownVideo = 4,
    bistAnalogueVideo = 5,
    bistDigitalVideo = 6,
    bistUnknown = 7,
    bistVendor = 128
} BdifImageSourceType;
```

Members

Members	Description
bistUnspecified = 0	Unspecified.
bistUnknownPhoto = 1	Unknown photo.
bistDigitalPhoto = 2	Digital photo.
bistScannedPhoto = 3	Scanned photo.
bistUnknownVideo = 4	Unknown video.
bistAnalogueVideo = 5	Analogue video.
bistDigitalVideo = 6	Digital video.
bistUnknown = 7	Unknown.
bistVendor = 128	Vendor.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bistUnspecified	-	+	-	-	-	+	-	-	+	-
bistUnknownPhoto	-	+	-	-	-	+	-	-	+	-
bistDigitalPhoto	-	+	-	-	-	+	-	-	+	-
bistScannedPhoto	-	+	-	-	-	+	-	-	+	-
bistUnknownVideo	-	+	-	-	-	+	-	-	+	-
bistAnalogueVideo	-	+	-	-	-	+	-	-	+	-
bistDigitalVideo	-	+	-	-	-	+	-	-	+	-
bistUnknown	-	+	-	-	-	+	-	-	+	-
bistVendor	-	+	-	-	-	+	-	-	+	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module ([see page 14](#))

6.1.1.2.1.15 BdifIrisOrientation Enumeration

Enumerates iris orientation constants.

C++

```
typedef enum BdifIrisOrientation_ {
    bioUndefined = 0,
    bioBase = 1,
    bioFlipped = 2
} BdifIrisOrientation;
```

Members

Members	Description
bioUndefined = 0	Undefined iris orientation.
bioBase = 1	Base iris orientation.
bioFlipped = 2	Flipped iris orientation.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bioUndefined	-	+	-	+	-	-	-	-	-	+

bioBase	-	+	-	+	-	-	-	-	-	+
bioFlipped	-	+	-	+	-	-	-	-	-	+

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.16 BdifIrisScanType Enumeration

Enumerates iris scan type constants.

C++

```
typedef enum BdifIrisScanType_ {
    bistUndefined = 0,
    bistProgressive = 1,
    bistInterlaceFrame = 2,
    bistInterlaceField = 3,
    bistCorrected = 4
} BdifIrisScanType;
```

Members

Members	Description
bistUndefined = 0	Undefined iris scan type.
bistProgressive = 1	Progressive scan type
bistInterlaceFrame = 2	Interlaced frame iris scan type.
bistInterlaceField = 3	Interlaced field iris scan type.
bistCorrected = 4	Corrected iris scan type.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bistUndefined	-	+	-	-	-	-	-	-	-	-
bistProgressive	-	+	-	+	-	-	-	-	-	+
bistInterlaceFrame	-	+	-	+	-	-	-	-	-	+
bistInterlaceField	-	+	-	+	-	-	-	-	-	+
bistCorrected	-	+	-	+	-	-	-	-	-	+

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (🔗 see page 14)

6.1.1.2.1.17 BdifScaleUnits Enumeration

Enumerates scale unit constants.

C++

```
typedef enum BdifScaleUnits_ {
    bsuNone = 0,
    bsuPixelsPerInch = 1,
    bsuPixelsPerCentimeter = 2
} BdifScaleUnits;
```

Members

Members	Description
bsuNone = 0	No scale units.
bsuPixelsPerInch = 1	Pixels per inch.
bsuPixelsPerCentimeter = 2	Pixels per centimeter.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bsuNone	+	+	-	-	-	-	-	-	-	-
bsuPixelsPerInch	+	-	-	-	+	-	-	+	-	-
bsuPixelsPerCentimeter	+	-	-	-	+	-	-	+	-	-

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9

- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.1.18 BdifStandard Enumeration

Enumerates standard constants.

C++

```
typedef enum BdifStandard_ {
    bsIso = 0,
    bsAnsi = 1
} BdifStandard;
```

Members

Members	Description
bsIso = 0	Iso standard.
bsAnsi = 1	Ansi standard.

Remarks

Members compatibility with various formats:

Memeber	1	2	3	4	5	6	7	8	9	10
bsIso	-	-	+	+	+	+	+	+	+	+
bsAnsi	-	-	+	+	+	+	+	+	+	+

List of standards and their corresponding numbers:

- ANSI/NIST-ITL 1-2000 - 1
- ANSI/NIST-ITL 1-2007 - 2
- ANSI INCITS 378-2004 - 3
- ANSI INCITS 379-2004 - 4
- ANSI INCITS 381-2004 - 5
- ANSI INCITS 385-2004 - 6
- ISO/IEC 19794-2:2005 - 7
- ISO/IEC 19794-4:2005 - 8
- ISO/IEC 19794-5:2005 - 9
- ISO/IEC 19794-6:2005 - 10

Module

BdifTypes Module (see page 14)

6.1.1.2.2 Macros

6.1.1.2.2.1 BDIF_DO_NOT_CHECK_CBEFF_PRODUCT_ID Macro

The flag which is used to indicate not check Cbeff product ID.

C++

```
#define BDIF_DO_NOT_CHECK_CBEFF_PRODUCT_ID 0x00000002
```

Module

BdifTypes Module ([see page 14](#))

6.1.1.2.3 Files**6.1.1.2.3.1 BdifTypes.h**

Header file for BdifTypes. Contains types and enumerations.

Enumerations

	Name	Description
	BdifEyeColor_ (see page 15)	Enumerates eye color constants.
	BdifEyePosition_ (see page 17)	Enumerates eye position constants.
	BdifFaceExpression_ (see page 17)	Enumerates facial feature constants.
	BdifFaceFeaturePointType_ (see page 19)	Enumerates face feature point type constants.
	BdifFaceProperties_ (see page 19)	Enumerates basic face properties constants.
	BdifFPatternClass_ (see page 21)	Enumerates fingerprint pattern class constants.
	BdifFPImpressionType_ (see page 22)	Enumerates fingerprint impression type constants.
	BdifFPMinutiaType_ (see page 25)	Enumerates fingerprint minutia type constants.
	BdifFPPosition_ (see page 26)	Enumerates fingerprint position constants.
	BdifGender_ (see page 28)	Enumerates gender constants.
	BdifHairColor_ (see page 29)	Enumerates hair color constants.
	BdifImageSourceType_ (see page 31)	Enumerates image source type constants.
	BdifIrisOrientation_ (see page 32)	Enumerates iris orientation constants.
	BdifIrisScanType_ (see page 33)	Enumerates iris scan type constants.
	BdifScaleUnits_ (see page 34)	Enumerates scale unit constants.
	BdifStandard_ (see page 35)	Enumerates standard constants.
	BdifEyeColor (see page 15)	Enumerates eye color constants.
	BdifEyePosition (see page 17)	Enumerates eye position constants.
	BdifFaceExpression (see page 17)	Enumerates facial feature constants.
	BdifFaceFeaturePointType (see page 19)	Enumerates face feature point type constants.
	BdifFaceProperties (see page 19)	Enumerates basic face properties constants.
	BdifFPatternClass (see page 21)	Enumerates fingerprint pattern class constants.
	BdifFPImpressionType (see page 22)	Enumerates fingerprint impression type constants.
	BdifFPMinutiaType (see page 25)	Enumerates fingerprint minutia type constants.
	BdifFPPosition (see page 26)	Enumerates fingerprint position constants.
	BdifGender (see page 28)	Enumerates gender constants.
	BdifHairColor (see page 29)	Enumerates hair color constants.
	BdifImageSourceType (see page 31)	Enumerates image source type constants.
	BdifIrisOrientation (see page 32)	Enumerates iris orientation constants.

	BdifIrisScanType (see page 33)	Enumerates iris scan type constants.
	BdifScaleUnits (see page 34)	Enumerates scale unit constants.
	BdifStandard (see page 35)	Enumerates standard constants.

Macros

Name	Description
BDIF_DO_NOT_CHECK_CBEFF_PRODUCT_ID (see page 35)	The flag which is used to indicate not check Cbeff product ID.

Module

BdifTypes Module (see page 14)

Structures

	Name	Description
	BdifFaceFeaturePoint_ (see page 18)	Enumerates face feature point constants.
	BdifFPMinutiaNeighbor_ (see page 25)	Enumerates fingerprint minutia type neighbors.
	BdifFaceFeaturePoint (see page 18)	Enumerates face feature point constants.
	BdifFPMinutiaNeighbor (see page 25)	Enumerates fingerprint minutia type neighbors.

6.1.1.3 CbeffBdbFormatIdentifiers Module

Specifies CBEFF Biometric Data Block (BDB) format identifiers registered with International Biometric Industry Association (IBIA).

Files

Name	Description
CbeffBdbFormatIdentifiers.h (see page 44)	Header file for Cbeff module. Specifies CBEFF Biometric Data Block (BDB) format identifiers registered with International Biometric Industry Association (IBIA).

Macros

Name	Description
CBEFF_BDBFI_ID3_SEMICONDUCTORS_COMPACT_CARD_EXTENSION (see page 41)	The CBEFF DBD format identifier assigned to id3 Semiconductors compact card extension format.

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FACE_IMAGE (see page 41)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics face-image format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_IMAGE (see page 41)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-image format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_N (see page 41)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-compact-n format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_V (see page 41)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-compact-v format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_N (see page 42)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-normal-n format.

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_V (see page 42)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-normal-v format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_N (see page 42)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-record-n format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_X (see page 42)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-record-x format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_DISCRETE_FOURIER_TRANSFORM (see page 43)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-discrete-fourier-transform format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_GABOR_FILTER (see page 43)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-gabor-filter format.

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_QUANTIZED_CO_SINUSOIDAL_TRIPLET (see page 43)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-quantized-cosinusoidal-triplet format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_POLAR (see page 43)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics iris-image-polar format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_RECTILINEAR (see page 44)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics iris-image-rectilinear format.
CBEFF_BDBFI_NEUROTECHNOLOGIJA_NLRECORD_3 (see page 44)	The CBEFF DBD format identifier assigned by Neurotehnologija for NLRecord version 3 format.
CBEFF_BDBFI_NEUROTECHNOLOGIJA_NTTEMPLATE (see page 44)	The CBEFF DBD format identifier assigned by Neurotehnologija for NTemplate.

6.1.1.3.1 Macros

6.1.1.3.1.1 CBEFF_BDBFI_ID3_SEMICONDUCTORS_COMPACT_CARD_EXTENSION Macro

The CBEFF DBD format identifier assigned to id3 Semiconductors compact card extension format.

C++

```
#define CBEFF_BDBFI_ID3_SEMICONDUCTORS_COMPACT_CARD_EXTENSION 0x0001
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.1.2 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FACE_IMAGE Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics face-image format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FACE_IMAGE 0x0008
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.1.3 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_IMAGE Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-image format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_IMAGE 0x0007
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.1.4

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_N Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-compact-n format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_N 0x0006
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.1.5

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_V Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-compact-v format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_V 0x0005
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.1.6

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_N

Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-normal-n format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_N 0x0004
```

Module

CbeffBdbFormatIdentifiers Module ([see page 37](#))

6.1.1.3.1.7

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_V

Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-normal-v format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_V 0x0003
```

Module

CbeffBdbFormatIdentifiers Module ([see page 37](#))

6.1.1.3.1.8

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_N

Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-record-n format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_N 0x0001
```

Module

CbeffBdbFormatIdentifiers Module ([see page 37](#))

6.1.1.3.1.9

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_X

Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-record-x format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_X 0x0002
```

Module

CbeffBdbFormatIdentifiers Module ([see page 37](#))

6.1.1.3.1.10**CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_DISCRETE_FOURIER_TRANSFORM****Macro**

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-discrete-fourier-transform format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_DISCRETE_FOURIER_TRANSFORM 0x000C
```

Module

CbeffBdbFormatIdentifiers Module ([see page 37](#))

6.1.1.3.1.11**CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_GABOR_FILTER****Macro**

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-gabor-filter format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_GABOR_FILTER 0x000D
```

Module

CbeffBdbFormatIdentifiers Module ([see page 37](#))

6.1.1.3.1.12**CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_QUANTIZED_CO_SINUSOIDAL_TRIPLET****Macro**

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-quantized-co-sinusoidal-triplet format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_QUANTIZED_CO_SINUSOIDAL_TRIPLET 0x000A
```

Module

CbeffBdbFormatIdentifiers Module ([see page 37](#))

6.1.1.3.1.13 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_POLAR**Macro**

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics iris-image-polar format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_POLAR 0x000B
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.1.14 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_RECTILINEAR Macro

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics iris-image-rectilinear format.

C++

```
#define CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_RECTILINEAR 0x0009
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.1.15 CBEFF_BDBFI_NEUROTECHNOLOGIJA_NLRECORD_3 Macro

The CBEFF DBD format identifier assigned by Neurotehnologija for NLRecord version 3 format.

C++

```
#define CBEFF_BDBFI_NEUROTECHNOLOGIJA_NLRECORD_3 0x0302
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.1.16 CBEFF_BDBFI_NEUROTECHNOLOGIJA_NTEMPLATE Macro

The CBEFF DBD format identifier assigned by Neurotehnologija for NTemplate.

C++

```
#define CBEFF_BDBFI_NEUROTECHNOLOGIJA_NTEMPLATE 0x0000
```

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.3.2 Files

6.1.1.3.2.1 CbeffBdbFormatIdentifiers.h

Header file for Cbeff module. Specifies CBEFF Biometric Data Block (BDB) format identifiers registered with International Biometric Industry Association (IBIA).

Macros

Name	Description
CBEFF_BDBFI_ID3_SEMICONDUCTORS_COMPACT_CARD_EXTENSION (see page 41)	The CBEFF DBD format identifier assigned to id3 Semiconductors compact card extension format.

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FACE_IMAGE (see page 41)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics face-image format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_IMAGE (see page 41)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-image format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_N (see page 41)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-compact-n format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_COMPACT_V (see page 41)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-compact-v format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_N (see page 42)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-normal-n format.

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_CARD_NORMAL_V (see page 42)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-card-normal-v format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_N (see page 42)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-record-n format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_MINUTIAE_RECORD_X (see page 42)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-minutiae-record-x format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_DISCRETE_FOURIER_TRANSFORM (see page 43)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-discrete-fourier-transform format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_GABOR_FILTER (see page 43)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-gabor-filter format.

CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FINGER_PATTERN_SPECTRAL_QUANTIZED_CO_SINUSOIDAL_TRIPLET (see page 43)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics finger-pattern-spectral-quantized-co-sinusoidal-triplet format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_POLAR (see page 43)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics iris-image-polar format.
CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS_IMAGE_RECTILINEAR (see page 44)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics iris-image-rectilinear format.
CBEFF_BDBFI_NEUROTECHNOLOGIJA_NLRECORD_3 (see page 44)	The CBEFF DBD format identifier assigned by Neurotehnologija for NLRecord version 3 format.
CBEFF_BDBFI_NEUROTECHNOLOGIJA_NTTEMPLATE (see page 44)	The CBEFF DBD format identifier assigned by Neurotehnologija for NTemplate.

Module

CbeffBdbFormatIdentifiers Module (see page 37)

6.1.1.4 CbeffBiometricOrganizations Module

Specifies CBEFF biometric organizations identifiers registered with International Biometric Industry Association (IBIA).

Files

Name	Description
CbeffBiometricOrganizations.h (see page 63)	Header file for CbeffBiometricOrganizations module. Specifies CBEFF biometric organizations identifiers registered with International Biometric Industry Association (IBIA).

Macros

Name	Description
CBEFF_BO_123ID (see page 51)	The CBEFF biometric organization identifier assigned to 123ID, Inc.
CBEFF_BO_A_4_VISION (see page 51)	The CBEFF biometric organization identifier assigned to A 4 Vision, S.A.
CBEFF_BO_ACSYS_BIOMETRICS (see page 51)	The CBEFF biometric organization identifier assigned to AcSys Biometrics Corp.
CBEFF_BO_ACTIVCARD (see page 51)	The CBEFF biometric organization identifier assigned to Activard.
CBEFF_BO_ANTHEUS_TECHNOLOGY (see page 51)	The CBEFF biometric organization identifier assigned to Antheus Technology.
CBEFF_BO_ATMEL (see page 51)	The CBEFF biometric organization identifier assigned to Atmel.
CBEFF_BO_AURORA_WIRELESS_TECHNOLOGIES (see page 52)	The CBEFF biometric organization identifier assigned to Aurora Wireless Technologies, Inc.
CBEFF_BO_AUTHEN_TEC (see page 52)	The CBEFF biometric organization identifier assigned to Authen Tec.
CBEFF_BO_AWARE (see page 52)	The CBEFF biometric organization identifier assigned to Aware, Inc.
CBEFF_BO_BIO_WISE (see page 52)	The CBEFF biometric organization identifier assigned to Bio Wise, NV.
CBEFF_BO_BIOKEY_INTERNATIONAL (see page 52)	The CBEFF biometric organization identifier assigned to BIO-key International.
CBEFF_BO_BIOKEY_INTERNATIONAL_2 (see page 52)	The CBEFF biometric organization identifier assigned to Bio-Key International.
CBEFF_BO_BIOLINK_TECHNOLOGIES_INTERNATIONAL (see page 53)	The CBEFF biometric organization identifier assigned to BioLink Technologies International, Inc.
CBEFF_BO_BIOSCRYPT (see page 53)	The CBEFF biometric organization identifier assigned to Bioscrypt, Inc.
CBEFF_BO_BIOVISION (see page 53)	The CBEFF biometric organization identifier assigned to Biovision, Inc.
CBEFF_BO_CYBER_SIGN (see page 53)	The CBEFF biometric organization identifier assigned to Cyber SIGN, Inc.
CBEFF_BO_COGENT_SYSTEMS (see page 53)	The CBEFF biometric organization identifier assigned to Cogent Systems, Inc.
CBEFF_BO_CRYPTOMETRICS (see page 54)	The CBEFF biometric organization identifier assigned to CryptoMetrics.
CBEFF_BO_CROSS_MATCH_TECHNOLOGIES (see page 54)	The CBEFF biometric organization identifier assigned to Cross Match Technologies, Inc.
CBEFF_BO_DERMALOG_IDENTIFICATION_SYSTEMS (see page 54)	The CBEFF biometric organization identifier assigned to DERMALOG Identification Systems, GmbH.

CBEFF_BO_DIGITALPERSONA (see page 54)	The CBEFF biometric organization identifier assigned to DigitalPersona, Inc.
CBEFF_BO_EAST_SHORE_TECHNOLOGIES (see page 54)	The CBEFF biometric organization identifier assigned to East Shore Technologies.
CBEFF_BO_ECRYPT (see page 54)	The CBEFF biometric organization identifier assigned to eCryp, Inc.
CBEFF_BO_FINGERPRINT_CARDS (see page 55)	The CBEFF biometric organization identifier assigned to Fingerprint Cards AB.
CBEFF_BO_GERMAN_INSTITUTE_FOR_STANDARDIZATION (see page 55)	The CBEFF biometric organization identifier assigned to German Institute for Standardization (Deutsches Institut für Normung E.V.-DIN).
CBEFF_BO_GREEN_BIT_AMERICAS (see page 55)	The CBEFF biometric organization identifier assigned to Green Bit Americas Inc.
CBEFF_BO_GRIAULE_TECNOLOGIA (see page 55)	The CBEFF biometric organization identifier assigned to Griaule Tecnologia LTDA.
CBEFF_BO_GUARDWARE_SYSTEMS (see page 55)	The CBEFF biometric organization identifier assigned to Guardware Systems, Ltd.
CBEFF_BO_HITACHI (see page 55)	The CBEFF biometric organization identifier assigned to Hitachi, LTD.
CBEFF_BO_HUMANSKAN (see page 56)	The CBEFF biometric organization identifier assigned to HumanScan GmbH.
CBEFF_BO_ID3_SEMICONDUCTORS (see page 56)	The CBEFF biometric organization identifier assigned to id3 Semiconductors.
CBEFF_BO_IDENTIX (see page 56)	The CBEFF biometric organization identifier assigned to Identix Corporation.
CBEFF_BO_IDENTIX_2 (see page 56)	The CBEFF biometric organization identifier assigned to Identix, Inc.
CBEFF_BO_IMAGE_WARE_SYSTEMS (see page 56)	The CBEFF biometric organization identifier assigned to Image Ware Systems, Inc.
CBEFF_BO_INCITS_TC_M1_BIOMETRICS (see page 57)	The CBEFF biometric organization identifier assigned to INCITS Technical Committee M1 - Biometrics.
CBEFF_BO_INFINEON_TECHNOLOGIES (see page 57)	The CBEFF biometric organization identifier assigned to Infineon Technologies, AG.
CBEFF_BO_INNOVATRICS (see page 57)	The CBEFF biometric organization identifier assigned to Innovatricks.
CBEFF_BO_INTERNATIONAL_BIOMETRIC_GROUP (see page 57)	The CBEFF biometric organization identifier assigned to International Biometric Group.
CBEFF_BO_INVESTORSHELPERS (see page 57)	The CBEFF biometric organization identifier assigned to InvestorsHelpers, Inc.
CBEFF_BO_IRIDIAN_TECHNOLOGIES (see page 57)	The CBEFF biometric organization identifier assigned to Iridian Technologies, Inc.
CBEFF_BO_ISO_IEC_JTC_1_SC_37_BIOMETRICS (see page 58)	The CBEFF biometric organization identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics.
CBEFF_BO_JANUS_ASSOCIATES (see page 58)	The CBEFF biometric organization identifier assigned to JANUS Associates, Inc.
CBEFF_BO_LG_ELECTRONICS_USA (see page 58)	The CBEFF biometric organization identifier assigned to LG Electronics USA, Inc.
CBEFF_BO_LOGICO_SMARTCARD_SOLUTIONS (see page 58)	The CBEFF biometric organization identifier assigned to LOGICO Smartcard Solutions, AG.
CBEFF_BO_LUMIDIGM (see page 58)	The CBEFF biometric organization identifier assigned to Lumidigm.
CBEFF_BO_NEC_SOLUTIONS_AMERICA (see page 58)	The CBEFF biometric organization identifier assigned to NEC Solutions America, Inc.

CBEFF_BO_NEUROTECHNOLOGIJA (see page 59)	The CBEFF biometric organization identifier assigned to Neurotehnologija.
CBEFF_BO_NIST (see page 59)	The CBEFF biometric organization identifier assigned to National Institute of Standards and Technology (NIST).
CBEFF_BO_NITGEN (see page 59)	The CBEFF biometric organization identifier assigned to NITGEN.
CBEFF_BO_NOT_FOR_USE (see page 59)	The CBEFF biometric organization identifier which is not allowed to be used by any of the biometric organizations.
CBEFF_BO_OMNIPERCEPTION (see page 59)	The CBEFF biometric organization identifier assigned to OmniPerception Limited.
CBEFF_BO_PRECISE_BIOMETRICS (see page 60)	The CBEFF biometric organization identifier assigned to Precise Biometrics, AB.
CBEFF_BO_PRIVATE (see page 60)	The CBEFF biometric organization identifier for private use (not uniquely assigned by IBIA).
CBEFF_BO_RECOGNITION_SYSTEMS (see page 60)	The CBEFF biometric organization identifier assigned to Recognition Systems, Inc.
CBEFF_BO_SAFLINK (see page 60)	The CBEFF biometric organization identifier assigned to SAFLINK Corporation.
CBEFF_BO_SAGEM_MORPHO (see page 60)	The CBEFF biometric organization identifier assigned to Sagem Morpho.
CBEFF_BO_SECUGEN (see page 60)	The CBEFF biometric organization identifier assigned to SecuGen Corporation.
CBEFF_BO_SECURE_DESIGN (see page 61)	The CBEFF biometric organization identifier assigned to Secure Design.
CBEFF_BO_SILEX_TECHNOLOGY (see page 61)	The CBEFF biometric organization identifier assigned to Silex Technology, Inc.
CBEFF_BO_SONDA_TECHNOLOGIES (see page 61)	The CBEFF biometric organization identifier assigned to Sonda Technologies.
CBEFF_BO_STARTEK_ENGINEERING (see page 61)	The CBEFF biometric organization identifier assigned to StarTek Engineering, Inc.
CBEFF_BO_SUPREMA (see page 61)	The CBEFF biometric organization identifier assigned to Suprema, Inc.
CBEFF_BO_TESTING_1 (see page 61)	The CBEFF biometric organization identifier reserved for testing.
CBEFF_BO_TESTING_2 (see page 62)	The CBEFF biometric organization identifier reserved for testing.
CBEFF_BO_THALES_IDENTIFICATION (see page 62)	The CBEFF biometric organization identifier assigned to Thales Identification.
CBEFF_BO_ULTRA_SCAN (see page 62)	The CBEFF biometric organization identifier assigned to Ultra-Scan Corporation.
CBEFF_BO_UPEK (see page 62)	The CBEFF biometric organization identifier assigned to UPEK, Inc.
CBEFF_BO_VALIDITY (see page 62)	The CBEFF biometric organization identifier assigned to Validity, Inc.
CBEFF_BO_VERIDICOM (see page 63)	The CBEFF biometric organization identifier assigned to Veridicom, Inc.
CBEFF_BO_VERIDT (see page 63)	The CBEFF biometric organization identifier assigned to Veridt.
CBEFF_BO_VIISAGE (see page 63)	The CBEFF biometric organization identifier assigned to Viisage.
CBEFF_BO_XTEC (see page 63)	The CBEFF biometric organization identifier assigned to XTec, Incorporated.

6.1.1.4.1 Macros

6.1.1.4.1.1 CBEFF_BO_123ID Macro

The CBEFF biometric organization identifier assigned to 123ID, Inc.

C++

```
#define CBEFF_BO_123ID 0x0038
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.2 CBEFF_BO_A_4_VISION Macro

The CBEFF biometric organization identifier assigned to A 4 Vision, S.A.

C++

```
#define CBEFF_BO_A_4_VISION 0x0010
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.3 CBEFF_BO_ACSYS_BIOMETRICS Macro

The CBEFF biometric organization identifier assigned to AcSys Biometrics Corp.

C++

```
#define CBEFF_BO_ACSYS_BIOMETRICS 0x0022
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.4 CBEFF_BO_ACTIVCARD Macro

The CBEFF biometric organization identifier assigned to Activard.

C++

```
#define CBEFF_BO_ACTIVCARD 0x0020
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.5 CBEFF_BO_ANTHEUS_TECHNOLOGY Macro

The CBEFF biometric organization identifier assigned to Antheus Technology.

C++

```
#define CBEFF_BO_ANTHEUS_TECHNOLOGY 0x0032
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.6 CBEFF_BO_ATMEL Macro

The CBEFF biometric organization identifier assigned to Atmel.

C++

```
#define CBEFF_BO_ATMEL 0x0041
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.7 CBEFF_BO_AURORA_WIRELESS_TECHNOLOGIES Macro

The CBEFF biometric organization identifier assigned to Aurora Wireless Technologies, Inc.

C++

```
#define CBEFF_BO_AURORA_WIRELESS_TECHNOLOGIES 0x0014
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.8 CBEFF_BO_AUTHEN_TEC Macro

The CBEFF biometric organization identifier assigned to Authen Tec.

C++

```
#define CBEFF_BO_AUTHEN_TEC 0x0042
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.9 CBEFF_BO_AWARE Macro

The CBEFF biometric organization identifier assigned to Aware, Inc.

C++

```
#define CBEFF_BO_AWARE 0x003B
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.10 CBEFF_BO_BIO_WISE Macro

The CBEFF biometric organization identifier assigned to Bio Wise, NV.

C++

```
#define CBEFF_BO_BIO_WISE 0x0024
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.11 CBEFF_BO_BIOKEY_INTERNATIONAL Macro

The CBEFF biometric organization identifier assigned to BIO-key International.

C++

```
#define CBEFF_BO_BIOKEY_INTERNATIONAL 0x0030
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.12 CBEFF_BO_BIOKEY_INTERNATIONAL_2 Macro

The CBEFF biometric organization identifier assigned to Bio-Key International.

C++

```
#define CBEFF_BO_BIOKEY_INTERNATIONAL_2 0x0039
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.13 CBEFF_BO_BIOLINK_TECHNOLOGIES_INTERNATIONAL Macro

The CBEFF biometric organization identifier assigned to BioLink Technologies International, Inc.

C++

```
#define CBEFF_BO_BIOLINK_TECHNOLOGIES_INTERNATIONAL 0x001F
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.14 CBEFF_BO_BIOSCRYPT Macro

The CBEFF biometric organization identifier assigned to Bioscrypt, Inc.

C++

```
#define CBEFF_BO_BIOSCRYPT 0x0002
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.15 CBEFF_BO_BIOVISION Macro

The CBEFF biometric organization identifier assigned to Biovision, Inc.

C++

```
#define CBEFF_BO_BIOVISION 0x0045
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.16 CBEFF_BO_CYBER_SIGN Macro

The CBEFF biometric organization identifier assigned to Cyber SIGN, Inc.

C++

```
#define CBEFF_BO_CYBER_SIGN 0x0007
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.17 CBEFF_BO_COGENT_SYSTEMS Macro

The CBEFF biometric organization identifier assigned to Cogent Systems, Inc.

C++

```
#define CBEFF_BO_COGENT_SYSTEMS 0x0017
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.18 CBEFF_BO_CRYPTOMETRICS Macro

The CBEFF biometric organization identifier assigned to CryptoMetrics.

C++

```
#define CBEFF_BO_CRYPTOMETRICS 0x002F
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.19 CBEFF_BO_CROSS_MATCH_TECHNOLOGIES Macro

The CBEFF biometric organization identifier assigned to Cross Match Technologies, Inc.

C++

```
#define CBEFF_BO_CROSS_MATCH_TECHNOLOGIES 0x0018
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.20 CBEFF_BO_DERMALOG_IDENTIFICATION_SYSTEMS Macro

The CBEFF biometric organization identifier assigned to DERMALOG Identification Systems, GmbH.

C++

```
#define CBEFF_BO_DERMALOG_IDENTIFICATION_SYSTEMS 0x000D
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.21 CBEFF_BO_DIGITALPERSONA Macro

The CBEFF biometric organization identifier assigned to DigitalPersona, Inc.

C++

```
#define CBEFF_BO_DIGITALPERSONA 0x0033
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.22 CBEFF_BO_EAST_SHORE_TECHNOLOGIES Macro

The CBEFF biometric organization identifier assigned to East Shore Technologies.

C++

```
#define CBEFF_BO_EAST_SHORE_TECHNOLOGIES 0x0036
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.23 CBEFF_BO_ECRYP Macro

The CBEFF biometric organization identifier assigned to eCryp, Inc.

C++

```
#define CBEFF_BO_ECRYP 0x0008
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.24 CBEFF_BO_FINGERPRINT_CARDS Macro

The CBEFF biometric organization identifier assigned to Fingerprint Cards AB.

C++

```
#define CBEFF_BO_FINGERPRINT_CARDS 0x0009
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.25 CBEFF_BO_GERMAN_INSTITUTE_FOR_STANDARDIZATION Macro

The CBEFF biometric organization identifier assigned to German Institute for Standardization (Deutsches Institut für Normung E.V.-DIN).

C++

```
#define CBEFF_BO_GERMAN_INSTITUTE_FOR_STANDARDIZATION 0x001A
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.26 CBEFF_BO_GREEN_BIT_AMERICAS Macro

The CBEFF biometric organization identifier assigned to Green Bit Americas Inc.

C++

```
#define CBEFF_BO_GREEN_BIT_AMERICAS 0x0040
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.27 CBEFF_BO_GRIAULE_TECNOLOGIA Macro

The CBEFF biometric organization identifier assigned to Griaule Tecnologia LTDA.

C++

```
#define CBEFF_BO_GRIAULE_TECNOLOGIA 0x003A
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.28 CBEFF_BO_GUARDWARE_SYSTEMS Macro

The CBEFF biometric organization identifier assigned to Guardware Systems, Ltd.

C++

```
#define CBEFF_BO_GUARDWARE_SYSTEMS 0x0026
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.29 CBEFF_BO_HITACHI Macro

The CBEFF biometric organization identifier assigned to Hitachi, LTD.

C++

```
#define CBEFF_BO_HITACHI 0x0037
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.30 CBEFF_BO_HUMANSKAN Macro

The CBEFF biometric organization identifier assigned to HumanScan GmbH.

C++

```
#define CBEFF_BO_HUMANSKAN 0x0021
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.31 CBEFF_BO_ID3_SEMICONDUCTORS Macro

The CBEFF biometric organization identifier assigned to id3 Semiconductors.

C++

```
#define CBEFF_BO_ID3_SEMICONDUCTORS 0x003F
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.32 CBEFF_BO_IDENTIX Macro

The CBEFF biometric organization identifier assigned to Identix Corporation.

C++

```
#define CBEFF_BO_IDENTIX 0x0003
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.33 CBEFF_BO_IDENTIX_2 Macro

The CBEFF biometric organization identifier assigned to Identix, Inc.

C++

```
#define CBEFF_BO_IDENTIX_2 0x000C
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.34 CBEFF_BO_IMAGE_WARE_SYSTEMS Macro

The CBEFF biometric organization identifier assigned to Image Ware Systems, Inc.

C++

```
#define CBEFF_BO_IMAGE_WARE_SYSTEMS 0x0043
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.35 CBEFF_BO_INCITS_TC_M1_BIOMETRICS Macro

The CBEFF biometric organization identifier assigned to INCITS Technical Committee M1 - Biometrics.

C++

```
#define CBEFF_BO_INCITS_TC_M1_BIOMETRICS 0x001B
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.36 CBEFF_BO_INFINEON_TECHNOLOGIES Macro

The CBEFF biometric organization identifier assigned to Infineon Technologies, AG.

C++

```
#define CBEFF_BO_INFINEON_TECHNOLOGIES 0x0004
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.37 CBEFF_BO_INNOVATRICS Macro

The CBEFF biometric organization identifier assigned to Innovatrics.

C++

```
#define CBEFF_BO_INNOVATRICS 0x0035
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.38 CBEFF_BO_INTERNATIONAL_BIOMETRIC_GROUP Macro

The CBEFF biometric organization identifier assigned to International Biometric Group.

C++

```
#define CBEFF_BO_INTERNATIONAL_BIOMETRIC_GROUP 0x0016
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.39 CBEFF_BO_INVESTORSHELPERS Macro

The CBEFF biometric organization identifier assigned to InvestorsHelpers, Inc.

C++

```
#define CBEFF_BO_INVESTORSHELPERS 0x0027
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.40 CBEFF_BO_IRIDIAN_TECHNOLOGIES Macro

The CBEFF biometric organization identifier assigned to Iridian Technologies, Inc.

C++

```
#define CBEFF_BO_IRIDIAN_TECHNOLOGIES 0x0005
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.41 CBEFF_BO_ISO_IEC_JTC_1_SC_37_BIOMETRICS Macro

The CBEFF biometric organization identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics.

C++

```
#define CBEFF_BO_ISO_IEC_JTC_1_SC_37_BIOMETRICS 0x0101
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.42 CBEFF_BO_JANUS_ASSOCIATES Macro

The CBEFF biometric organization identifier assigned to JANUS Associates, Inc.

C++

```
#define CBEFF_BO_JANUS_ASSOCIATES 0x002C
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.43 CBEFF_BO_LG_ELECTRONICS_USA Macro

The CBEFF biometric organization identifier assigned to LG Electronics USA, Inc.

C++

```
#define CBEFF_BO_LG_ELECTRONICS_USA 0x002A
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.44 CBEFF_BO_LOGICO_SMARTCARD_SOLUTIONS Macro

The CBEFF biometric organization identifier assigned to LOGICO Smartcard Solutions, AG.

C++

```
#define CBEFF_BO_LOGICO_SMARTCARD_SOLUTIONS 0x000E
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.45 CBEFF_BO_LUMIDIGM Macro

The CBEFF biometric organization identifier assigned to Lumidigm.

C++

```
#define CBEFF_BO_LUMIDIGM 0x0025
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.46 CBEFF_BO_NEC_SOLUTIONS_AMERICA Macro

The CBEFF biometric organization identifier assigned to NEC Solutions America, Inc.

C++

```
#define CBEFF_BO_NEC_SOLUTIONS_AMERICA 0x0011
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.47 CBEFF_BO_NEUROTECHNOLOGIJA Macro

The CBEFF biometric organization identifier assigned to Neurotehnologija.

C++

```
#define CBEFF_BO_NEUROTECHNOLOGIJA 0x0031
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.48 CBEFF_BO_NIST Macro

The CBEFF biometric organization identifier assigned to National Institute of Standards and Technology (NIST).

C++

```
#define CBEFF_BO_NIST 0x000F
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.49 CBEFF_BO_NITGEN Macro

The CBEFF biometric organization identifier assigned to NITGEN.

C++

```
#define CBEFF_BO_NITGEN 0x001C
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.50 CBEFF_BO_NOT_FOR_USE Macro

The CBEFF biometric organization identifier which is not allowed to be used by any of the biometric organizations.

C++

```
#define CBEFF_BO_NOT_FOR_USE 0xFFFF
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.51 CBEFF_BO_OMNIPERCEPTION Macro

The CBEFF biometric organization identifier assigned to OmniPerception Limited.

C++

```
#define CBEFF_BO_OMNIPERCEPTION 0x002D
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.52 CBEFF_BO_PRECISE_BIOMETRICS Macro

The CBEFF biometric organization identifier assigned to Precise Biometrics, AB.

C++

```
#define CBEFF_BO_PRECISE_BIOMETRICS 0x000B
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.53 CBEFF_BO_PRIVATE Macro

The CBEFF biometric organization identifier for private use (not uniquely assigned by IBIA).

C++

```
#define CBEFF_BO_PRIVATE 0xFEFE
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.54 CBEFF_BO_RECOGNITION_SYSTEMS Macro

The CBEFF biometric organization identifier assigned to Recognition Systems, Inc.

C++

```
#define CBEFF_BO_RECOGNITION_SYSTEMS 0x0019
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.55 CBEFF_BO_SAFLINK Macro

The CBEFF biometric organization identifier assigned to SAFLINK Corporation.

C++

```
#define CBEFF_BO_SAFLINK 0x0001
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.56 CBEFF_BO_SAGEM_MORPHO Macro

The CBEFF biometric organization identifier assigned to Sagem Morpho.

C++

```
#define CBEFF_BO_SAGEM_MORPHO 0x001D
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.57 CBEFF_BO_SECUGEN Macro

The CBEFF biometric organization identifier assigned to SecuGen Corporation.

C++

```
#define CBEFF_BO_SECUGEN 0x000A
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.58 CBEFF_BO_SECURE_DESIGN Macro

The CBEFF biometric organization identifier assigned to Secure Design.

C++

```
#define CBEFF_BO_SECURE_DESIGN 0x003D
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.59 CBEFF_BO_SILEX_TECHNOLOGY Macro

The CBEFF biometric organization identifier assigned to Silex Technology, Inc.

C++

```
#define CBEFF_BO_SILEX_TECHNOLOGY 0x0023
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.60 CBEFF_BO_SONDA_TECHNOLOGIES Macro

The CBEFF biometric organization identifier assigned to Sonda Technologies.

C++

```
#define CBEFF_BO_SONDA_TECHNOLOGIES 0x003C
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.61 CBEFF_BO_STARTEK_ENGINEERING Macro

The CBEFF biometric organization identifier assigned to StarTek Engineering, Inc.

C++

```
#define CBEFF_BO_STARTEK_ENGINEERING 0x002B
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.62 CBEFF_BO_SUPREMA Macro

The CBEFF biometric organization identifier assigned to Suprema, Inc.

C++

```
#define CBEFF_BO_SUPREMA 0x0044
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.63 CBEFF_BO_TESTING_1 Macro

The CBEFF biometric organization identifier reserved for testing.

C++

```
#define CBEFF_BO_TESTING_1 0xFFFF0
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.64 CBEFF_BO_TESTING_2 Macro

The CBEFF biometric organization identifier reserved for testing.

C++

```
#define CBEFF_BO_TESTING_2 0xFFFFE
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.65 CBEFF_BO_THALES_IDENTIFICATION Macro

The CBEFF biometric organization identifier assigned to Thales Identification.

C++

```
#define CBEFF_BO_THALES_IDENTIFICATION 0x0015
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.66 CBEFF_BO_ULTRA_SCAN Macro

The CBEFF biometric organization identifier assigned to Ultra-Scan Corporation.

C++

```
#define CBEFF_BO_ULTRA_SCAN 0x0013
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.67 CBEFF_BO_UPEK Macro

The CBEFF biometric organization identifier assigned to UPEK, Inc.

C++

```
#define CBEFF_BO_UPEK 0x0012
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.68 CBEFF_BO_VALIDITY Macro

The CBEFF biometric organization identifier assigned to Validity, Inc.

C++

```
#define CBEFF_BO_VALIDITY 0x0028
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.69 CBEFF_BO_VERIDICOM Macro

The CBEFF biometric organization identifier assigned to Veridicom, Inc.

C++

```
#define CBEFF_BO_VERIDICOM 0x0006
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.70 CBEFF_BO_VERIDT Macro

The CBEFF biometric organization identifier assigned to Veridt.

C++

```
#define CBEFF_BO_VERIDT 0x003E
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.71 CBEFF_BO_VIISAGE Macro

The CBEFF biometric organization identifier assigned to Viisage.

C++

```
#define CBEFF_BO_VIISAGE 0x0029
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.1.72 CBEFF_BO_XTEC Macro

The CBEFF biometric organization identifier assigned to XTec, Incorporated.

C++

```
#define CBEFF_BO_XTEC 0x0034
```

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.4.2 Files

6.1.1.4.2.1 CbeffBiometricOrganizations.h

Header file for CbeffBiometricOrganizations (see page 48) module. Specifies CBEFF biometric organizations identifiers registered with International Biometric Industry Association (IBIA).

Macros

Name	Description
CBEFF_BO_123ID (see page 51)	The CBEFF biometric organization identifier assigned to 123ID, Inc.
CBEFF_BO_A_4_VISION (see page 51)	The CBEFF biometric organization identifier assigned to A 4 Vision, S.A.
CBEFF_BO_ACSYS_BIOMETRICS (see page 51)	The CBEFF biometric organization identifier assigned to AcSys Biometrics Corp.

CBEFF_BO_ACTIVCARD (see page 51)	The CBEFF biometric organization identifier assigned to Activard.
CBEFF_BO_ANTHEUS_TECHNOLOGY (see page 51)	The CBEFF biometric organization identifier assigned to Antheus Technology.
CBEFF_BO_ATMEL (see page 51)	The CBEFF biometric organization identifier assigned to Atmel.
CBEFF_BO_AURORA_WIRELESS_TECHNOLOGIES (see page 52)	The CBEFF biometric organization identifier assigned to Aurora Wireless Technologies, Inc.
CBEFF_BO_AUTHEN_TEC (see page 52)	The CBEFF biometric organization identifier assigned to Authen Tec.
CBEFF_BO_AWARE (see page 52)	The CBEFF biometric organization identifier assigned to Aware, Inc.
CBEFF_BO_BIO_WISE (see page 52)	The CBEFF biometric organization identifier assigned to Bio Wise, NV.
CBEFF_BO_BIOKEY_INTERNATIONAL (see page 52)	The CBEFF biometric organization identifier assigned to BIO-key International.
CBEFF_BO_BIOKEY_INTERNATIONAL_2 (see page 52)	The CBEFF biometric organization identifier assigned to Bio-Key International.
CBEFF_BO_BIOLINK_TECHNOLOGIES_INTERNATIONAL (see page 53)	The CBEFF biometric organization identifier assigned to BioLink Technologies International, Inc.
CBEFF_BO_BIOSCRYPT (see page 53)	The CBEFF biometric organization identifier assigned to Bioscrypt, Inc.
CBEFF_BO_BIOVISION (see page 53)	The CBEFF biometric organization identifier assigned to Biovision, Inc.
CBEFF_BO_CYBER_SIGN (see page 53)	The CBEFF biometric organization identifier assigned to Cyber SIGN, Inc.
CBEFF_BO_COGENT_SYSTEMS (see page 53)	The CBEFF biometric organization identifier assigned to Cogent Systems, Inc.
CBEFF_BO_CRYPTOMETRICS (see page 54)	The CBEFF biometric organization identifier assigned to CryptoMetrics.
CBEFF_BO_CROSS_MATCH_TECHNOLOGIES (see page 54)	The CBEFF biometric organization identifier assigned to Cross Match Technologies, Inc.
CBEFF_BO_DERMALOG_IDENTIFICATION_SYSTEMS (see page 54)	The CBEFF biometric organization identifier assigned to DERMALOG Identification Systems, GmbH.
CBEFF_BO_DIGITALPERSONA (see page 54)	The CBEFF biometric organization identifier assigned to DigitalPersona, Inc.
CBEFF_BO_EAST_SHORE_TECHNOLOGIES (see page 54)	The CBEFF biometric organization identifier assigned to East Shore Technologies.
CBEFF_BO_ECRYP (see page 54)	The CBEFF biometric organization identifier assigned to eCryp, Inc.
CBEFF_BO_FINGERPRINT_CARDS (see page 55)	The CBEFF biometric organization identifier assigned to Fingerprint Cards AB.
CBEFF_BO_GERMAN_INSTITUTE_FOR_STANDARDIZATION (see page 55)	The CBEFF biometric organization identifier assigned to German Institute for Standardization (Deutsches Institut für Normung E.V.-DIN).
CBEFF_BO_GREEN_BIT_AMERICAS (see page 55)	The CBEFF biometric organization identifier assigned to Green Bit Americas Inc.
CBEFF_BO_GRIAULE_TECNOLOGIA (see page 55)	The CBEFF biometric organization identifier assigned to Griaule Tecnologia LTDA.
CBEFF_BO_GUARDWARE_SYSTEMS (see page 55)	The CBEFF biometric organization identifier assigned to Guardware Systems, Ltd.
CBEFF_BO_HITACHI (see page 55)	The CBEFF biometric organization identifier assigned to Hitachi, LTD.

CBEFF_BO_HUMANSKAN (see page 56)	The CBEFF biometric organization identifier assigned to HumanScan GmbH.
CBEFF_BO_ID3_SEMICONDUCTORS (see page 56)	The CBEFF biometric organization identifier assigned to id3 Semiconductors.
CBEFF_BO_IDENTIX (see page 56)	The CBEFF biometric organization identifier assigned to Identix Corporation.
CBEFF_BO_IDENTIX_2 (see page 56)	The CBEFF biometric organization identifier assigned to Identix, Inc.
CBEFF_BO_IMAGE_WARE_SYSTEMS (see page 56)	The CBEFF biometric organization identifier assigned to Image Ware Systems, Inc.
CBEFF_BO_INCITS_TC_M1_BIOMETRICS (see page 57)	The CBEFF biometric organization identifier assigned to INCITS Technical Committee M1 - Biometrics.
CBEFF_BO_INFINEON_TECHNOLOGIES (see page 57)	The CBEFF biometric organization identifier assigned to Infineon Technologies, AG.
CBEFF_BO_INNOVATRICS (see page 57)	The CBEFF biometric organization identifier assigned to Innovatricks.
CBEFF_BO_INTERNATIONAL_BIOMETRIC_GROUP (see page 57)	The CBEFF biometric organization identifier assigned to International Biometric Group.
CBEFF_BO_INVESTORSHELPERS (see page 57)	The CBEFF biometric organization identifier assigned to InvestorsHelpers, Inc.
CBEFF_BO_IRIDIAN_TECHNOLOGIES (see page 57)	The CBEFF biometric organization identifier assigned to Iridian Technologies, Inc.
CBEFF_BO_ISO_IEC_JTC_1_SC_37_BIOMETRICS (see page 58)	The CBEFF biometric organization identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics.
CBEFF_BO_JANUS_ASSOCIATES (see page 58)	The CBEFF biometric organization identifier assigned to JANUS Associates, Inc.
CBEFF_BO_LG_ELECTRONICS_USA (see page 58)	The CBEFF biometric organization identifier assigned to LG Electronics USA, Inc.
CBEFF_BO_LOGICO_SMARTCARD_SOLUTIONS (see page 58)	The CBEFF biometric organization identifier assigned to LOGICO Smartcard Solutions, AG.
CBEFF_BO_LUMIDIGM (see page 58)	The CBEFF biometric organization identifier assigned to Lumidigm.
CBEFF_BO_NEC_SOLUTIONS_AMERICA (see page 58)	The CBEFF biometric organization identifier assigned to NEC Solutions America, Inc.
CBEFF_BO_NEUROTECHNOLOGIJA (see page 59)	The CBEFF biometric organization identifier assigned to Neurotehnologija.
CBEFF_BO_NIST (see page 59)	The CBEFF biometric organization identifier assigned to National Institute of Standards and Technology (NIST).
CBEFF_BO_NITGEN (see page 59)	The CBEFF biometric organization identifier assigned to NITGEN.
CBEFF_BO_NOT_FOR_USE (see page 59)	The CBEFF biometric organization identifier which is not allowed to be used by any of the biometric organizations.
CBEFF_BO_OMNIPERCEPTION (see page 59)	The CBEFF biometric organization identifier assigned to OmniPerception Limited.
CBEFF_BO_PRECISE_BIOMETRICS (see page 60)	The CBEFF biometric organization identifier assigned to Precise Biometrics, AB.
CBEFF_BO_PRIVATE (see page 60)	The CBEFF biometric organization identifier for private use (not uniquely assigned by IBIA).
CBEFF_BO_RECOGNITION_SYSTEMS (see page 60)	The CBEFF biometric organization identifier assigned to Recognition Systems, Inc.
CBEFF_BO_SAFLINK (see page 60)	The CBEFF biometric organization identifier assigned to SAFLINK Corporation.
CBEFF_BO_SAGEM_MORPHO (see page 60)	The CBEFF biometric organization identifier assigned to Sagem Morpho.

CBEFF_BO_SECUGEN (see page 60)	The CBEFF biometric organization identifier assigned to SecuGen Corporation.
CBEFF_BO_SECURE_DESIGN (see page 61)	The CBEFF biometric organization identifier assigned to Secure Design.
CBEFF_BO_SILEX_TECHNOLOGY (see page 61)	The CBEFF biometric organization identifier assigned to Silex Technology, Inc.
CBEFF_BO_SONDA_TECHNOLOGIES (see page 61)	The CBEFF biometric organization identifier assigned to Sonda Technologies.
CBEFF_BO_STARTEK_ENGINEERING (see page 61)	The CBEFF biometric organization identifier assigned to StarTek Engineering, Inc.
CBEFF_BO_SUPREMA (see page 61)	The CBEFF biometric organization identifier assigned to Suprema, Inc.
CBEFF_BO_TESTING_1 (see page 61)	The CBEFF biometric organization identifier reserved for testing.
CBEFF_BO_TESTING_2 (see page 62)	The CBEFF biometric organization identifier reserved for testing.
CBEFF_BO_THALES_IDENTIFICATION (see page 62)	The CBEFF biometric organization identifier assigned to Thales Identification.
CBEFF_BO_ULTRA_SCAN (see page 62)	The CBEFF biometric organization identifier assigned to Ultra-Scan Corporation.
CBEFF_BO_UPEK (see page 62)	The CBEFF biometric organization identifier assigned to UPEK, Inc.
CBEFF_BO_VALIDITY (see page 62)	The CBEFF biometric organization identifier assigned to Validity, Inc.
CBEFF_BO_VERIDICOM (see page 63)	The CBEFF biometric organization identifier assigned to Veridicom, Inc.
CBEFF_BO_VERIDT (see page 63)	The CBEFF biometric organization identifier assigned to Veridt.
CBEFF_BO_VIISAGE (see page 63)	The CBEFF biometric organization identifier assigned to Viisage.
CBEFF_BO_XTEC (see page 63)	The CBEFF biometric organization identifier assigned to XTec, Incorporated.

Module

CbeffBiometricOrganizations Module (see page 48)

6.1.1.5 FMCard Module

Provides functionality for converting ISO/IEC 19794-2:2005 card format standard templates to and from Neurotechnology Finger Records (NFRecords).

Files

Name	Description
FMCard.h (see page 72)	Header file FMCard. Provides functionality for converting ISO/IEC 19794-2:2005 card format standard templates to and from Neurotechnology Finger Records (NFRecords).

Functions

	Name	Description
⇒	FMCardWriteCompactSizeMinutiae (see page 67)	Converts NFRecord to compact size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.
⇒	FMCardWriteNormalSizeMinutiae (see page 68)	Converts NFRecord to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

Macros

Name	Description
FMC_BDT_TAG_CELL_QUALITY_DATA (see page 70)	Tag for storing cell quality data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BDT_TAG_CORE_POINT_DATA (see page 70)	Tag for storing core point data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BDT_TAG_DELTA_POINT_DATA (see page 71)	Tag for storing delta point data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BDT_TAG_FINGER_MINUTIAE_DATA (see page 71)	Tag for storing finger minutiae data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BDT_TAG_RIDGE_COUNT_DATA (see page 71)	Tag for storing ridge count data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BIT_BHT_BAPT_TAG_FEATURE_HANDLING_INDICATOR (see page 71)	Tag for storing FmcFeatureHandling (see page 69) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
FMC_BIT_BHT_BAPT_TAG_MINUTIAE_ORDER (see page 71)	Tag for storing FmcMinutiaOrder (see page 69) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
FMC_BIT_BHT_BAPT_TAG_NUMBER_OF_MINUTIAE (see page 71)	Tag for storing minimal and maximal number of minutiae in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
FMC_DEFAULT_MAX_ENROLL_MC (see page 72)	Default maximal minutia count for enrollment.
FMC_DEFAULT_MAX_VERIFY_MC (see page 72)	Default maximal minutia count for verification.
FMC_DEFAULT_MIN_ENROLL_MC (see page 72)	Default minimal minutia count for enrollment.
FMC_DEFAULT_MIN_VERIFY_MC (see page 72)	Default minimal minutia count for verification.

Structs, Records, Enums

	Name	Description
	FmcFeatureHandling_ (see page 69)	Specifies additional fingerprint features that are supported by the card.
	FmcMinutiaOrder_ (see page 69)	Specifies minutia ordering required by the card.
	FmcFeatureHandling (see page 69)	Specifies additional fingerprint features that are supported by the card.
	FmcMinutiaOrder (see page 69)	Specifies minutia ordering required by the card.

6.1.1.5.1 Functions**6.1.1.5.1.1 FMCardWriteCompactSizeMinutiae Function**

Converts NFRecord to compact size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C++

```
NResult N_API FMCardWriteCompactSizeMinutiae(HNFRecord hNFRecord, FmcMinutiaOrder order,
void * pBuffer, NSizeType bufferSize, NUInt flags, NSizeType * pSize);
```

Parameters

Parameters	Description
HNRecord hNFRecord	[in] Handle to the NFRecord object.
FmcMinutiaOrder order	Specifies order to write minutiae in.

void * pBuffer	[out] Pointer to memory buffer to write minutiae in. Can be NULL (see page 133).
NSizeType bufferSize	Size of memory buffer to write minutiae in.
NUInt flags	Reserved, must be zero.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives size of written minutiae.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pBuffer is NULL (see page 133) and bufferSize is not zero or pHNFRecord is NULL (see page 133).

Remarks

If the FmcMinutiaOrder (see page 69) is specified in FMCardWriteCompactSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.fmcmoPolar example:

```
FMCardWriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.fmcmoPolar |
FmcMinutiaOrder.fmcmoAscending);
```

or

```
FMCardWriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.fmcmoPolar |
FmcMinutiaOrder.fmcmoDescending);
```

Only for FmcMinutiaOrder.fmcmoNone and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
FMCardWriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.fmcmoNone);
FMCardWriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.XOrYCoordinateExtension);
```

Module

FMCard Module (see page 66)

6.1.1.5.1.2 FMCardWriteNormalSizeMinutiae Function

Converts NFRecord to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C++

```
NResult N_API FMCardWriteNormalSizeMinutiae(HNFRecord hNFRecord, FmcMinutiaOrder order,
void * pBuffer, NSizeType bufferSize, NUInt flags, NSizeType * pSize);
```

Parameters

Parameters	Description
HNFRecord hNFRecord	[in] Handle to the NFRecord object.
FmcMinutiaOrder order	Specifies order to write minutiae in.
void * pBuffer	[out] Pointer to memory buffer to write minutiae in. Can be NULL (see page 133).
NSizeType bufferSize	Size of memory buffer to write minutiae in.
NUInt flags	Reserved, must be zero.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives size of written minutiae.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pBuffer</i> is NULL (see page 133) and <i>bufferSize</i> is not zero or <i>pHNFRecord</i> is NULL (see page 133).

Remarks

If the `FmcMinutiaOrder` ([see page 69](#)) is specified in `FMCardWriteNormalSizeMinutiae`, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using `FmcMinutiaOrder.fmcmoPolar` example:

```
FMCardWriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.fmcmoPolar |  
FmcMinutiaOrder.fmcmoAscending);
```

or

```
FMCardWriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.fmcmoPolar |  
FmcMinutiaOrder.fmcmoDescending);
```

Only for `FmcMinutiaOrder.fmcmoNone` and `FmcMinutiaOrder.XORyCoordinateExtension` sorting is not needed:

```
FMCardWriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.fmcmoNone);  
FMCardWriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.XORyCoordinateExtension);
```

Module

FMCard Module ([see page 66](#))

6.1.1.5.2 Structs, Records, Enums

6.1.1.5.2.1 FmcFeatureHandling Enumeration

Specifies additional fingerprint features that are supported by the card.

C++

```
typedef enum FmcFeatureHandling_ {  
    fmcfhNone = 0,  
    fmcfhRidgeCounts = 0x01,  
    fmcfhCorePoints = 0x02,  
    fmcfhDeltaPoints = 0x04,  
    fmcfhCellQuality = 0x08  
} FmcFeatureHandling;
```

Members

Members	Description
<code>fmcfhNone = 0</code>	None additional features are supported.
<code>fmcfhRidgeCounts = 0x01</code>	Ridge counts supported.
<code>fmcfhCorePoints = 0x02</code>	Core points supported.
<code>fmcfhDeltaPoints = 0x04</code>	Delta points supported.
<code>fmcfhCellQuality = 0x08</code>	Cell quality supported.

Module

FMCard Module ([see page 66](#))

6.1.1.5.2.2 FmcMinutiaOrder Enumeration

Specifies minutia ordering required by the card.

C++

```
typedef enum FmcMinutiaOrder_ {
    fmcmoNone = 0,
    fmcmoAscending = 0x01,
    fmcmoDescending = 0x02,
    fmcmoCartesianXY = 0x04,
    fmcmoCartesianYX = 0x08,
    fmcmoAngle = 0x0C,
    fmcmoPolar = 0x10,
    fmcmoXOrYCoordinateExtension = 0x20
} FmcMinutiaOrder;
```

Members

Members	Description
fmcmoNone = 0	Specifies that no ordering is required.
fmcmoAscending = 0x01	Specifies that minutiae are sorted ascending by the specified order.
fmcmoDescending = 0x02	Specifies that minutiae are sorted descending by the specified order.
fmcmoCartesianXY = 0x04	Specifies that minutiae are sorted by X coordinate. If X coordinates of two minutiae are equal Y coordinate is compared.
fmcmoCartesianYX = 0x08	Specifies that minutiae are sorted by Y coordinate. If Y coordinates of two minutiae are equal X coordinate is compared.
fmcmoAngle = 0x0C	Specifies that minutiae are sorted by minutia angle.
fmcmoPolar = 0x10	Specifies that minutiae are sorted by distance from minutiae center of mass. If distance of two minutiae are equal minutia angle is compared.
fmcmoXOrYCoordinateExtension = 0x20	Specifies that minutiae X or Y coordinate extension is used for compact format.

Module

FMCard Module (see page 66)

6.1.1.5.3 Macros**6.1.1.5.3.1 FMC_BDT_TAG_CELL_QUALITY_DATA Macro**

Tag for storing cell quality data in ISO/IEC 7816-11:2004 Biometric data template.

C++

```
#define FMC_BDT_TAG_CELL_QUALITY_DATA 0x94
```

Module

FMCard Module (see page 66)

6.1.1.5.3.2 FMC_BDT_TAG_CORE_POINT_DATA Macro

Tag for storing core point data in ISO/IEC 7816-11:2004 Biometric data template.

C++

```
#define FMC_BDT_TAG_CORE_POINT_DATA 0x92
```

Module

FMCard Module (see page 66)

6.1.1.5.3.3 FMC_BDT_TAG_DELTA_POINT_DATA Macro

Tag for storing delta point data in ISO/IEC 7816-11:2004 Biometric data template.

C++

```
#define FMC_BDT_TAG_DELTA_POINT_DATA 0x93
```

Module

FMCard Module (see page 66)

6.1.1.5.3.4 FMC_BDT_TAG_FINGER_MINUTIAE_DATA Macro

Tag for storing finger minutiae data in ISO/IEC 7816-11:2004 Biometric data template.

C++

```
#define FMC_BDT_TAG_FINGER_MINUTIAE_DATA 0x90
```

Module

FMCard Module (see page 66)

6.1.1.5.3.5 FMC_BDT_TAG_RIDGE_COUNT_DATA Macro

Tag for storing ridge count data in ISO/IEC 7816-11:2004 Biometric data template.

C++

```
#define FMC_BDT_TAG_RIDGE_COUNT_DATA 0x91
```

Module

FMCard Module (see page 66)

6.1.1.5.3.6 FMC_BIT_BHT_BAPT_TAG_FEATURE_HANDLING_INDICATOR Macro

Tag for storing FmcFeatureHandling (see page 69) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.

C++

```
#define FMC_BIT_BHT_BAPT_TAG_FEATURE_HANDLING_INDICATOR 0x83
```

Module

FMCard Module (see page 66)

6.1.1.5.3.7 FMC_BIT_BHT_BAPT_TAG_MINUTIAE_ORDER Macro

Tag for storing FmcMinutiaOrder (see page 69) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.

C++

```
#define FMC_BIT_BHT_BAPT_TAG_MINUTIAE_ORDER 0x82
```

Module

FMCard Module (see page 66)

6.1.1.5.3.8 FMC_BIT_BHT_BAPT_TAG_NUMBER_OF_MINUTIAE Macro

Tag for storing minimal and maximal number of minutiae in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.

C++

```
#define FMC_BIT_BHT_BAPT_TAG_NUMBER_OF_MINUTIAE 0x81
```

Module

FMCard Module (see page 66)

6.1.1.5.3.9 FMC_DEFAULT_MAX_ENROLL_MC Macro

Default maximal minutia count for enrollment.

C++

```
#define FMC_DEFAULT_MAX_ENROLL_MC 60
```

Module

FMCard Module (see page 66)

6.1.1.5.3.10 FMC_DEFAULT_MAX_VERIFY_MC Macro

Default maximal minutia count for verification.

C++

```
#define FMC_DEFAULT_MAX_VERIFY_MC 60
```

Module

FMCard Module (see page 66)

6.1.1.5.3.11 FMC_DEFAULT_MIN_ENROLL_MC Macro

Default minimal minutia count for enrollment.

C++

```
#define FMC_DEFAULT_MIN_ENROLL_MC 16
```

Module

FMCard Module (see page 66)

6.1.1.5.3.12 FMC_DEFAULT_MIN_VERIFY_MC Macro

Default minimal minutia count for verification.

C++

```
#define FMC_DEFAULT_MIN_VERIFY_MC 12
```

Module

FMCard Module (see page 66)

6.1.1.5.4 Files**6.1.1.5.4.1 FMCard.h**

Header file FMCard. Provides functionality for converting ISO/IEC 19794-2:2005 card format standard templates to and from Neurotechnology Finger Records (NFRecords).

Enumerations

	Name	Description
	FmcFeatureHandling_ (see page 69)	Specifies additional fingerprint features that are supported by the card.
	FmcMinutiaOrder_ (see page 69)	Specifies minutia ordering required by the card.
	FmcFeatureHandling (see page 69)	Specifies additional fingerprint features that are supported by the card.
	FmcMinutiaOrder (see page 69)	Specifies minutia ordering required by the card.

Functions

	Name	Description
	FMCardWriteCompactSizeMinutiae (see page 67)	Converts NFRecord to compact size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.
	FMCardWriteNormalSizeMinutiae (see page 68)	Converts NFRecord to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

Macros

Name	Description
FMC_BDT_TAG_CELL_QUALITY_DATA (see page 70)	Tag for storing cell quality data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BDT_TAG_CORE_POINT_DATA (see page 70)	Tag for storing core point data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BDT_TAG_DELTA_POINT_DATA (see page 71)	Tag for storing delta point data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BDT_TAG_FINGER_MINUTIAE_DATA (see page 71)	Tag for storing finger minutiae data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BDT_TAG_RIDGE_COUNT_DATA (see page 71)	Tag for storing ridge count data in ISO/IEC 7816-11:2004 Biometric data template.
FMC_BIT_BHT_BAPT_TAG_FEATURE_HANDLING_INDICATOR (see page 71)	Tag for storing FmcFeatureHandling (see page 69) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
FMC_BIT_BHT_BAPT_TAG_MINUTIAE_ORDER (see page 71)	Tag for storing FmcMinutiaOrder (see page 69) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
FMC_BIT_BHT_BAPT_TAG_NUMBER_OF_MINUTIAE (see page 71)	Tag for storing minimal and maximal number of minutiae in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
FMC_DEFAULT_MAX_ENROLL_MC (see page 72)	Default maximal minutia count for enrollment.
FMC_DEFAULT_MAX_VERIFY_MC (see page 72)	Default maximal minutia count for verification.
FMC_DEFAULT_MIN_ENROLL_MC (see page 72)	Default minimal minutia count for enrollment.
FMC_DEFAULT_MIN_VERIFY_MC (see page 72)	Default minimal minutia count for verification.

Module

FMCard Module ([see page 66](#))

6.1.1.6 NBiometricStandards Module

Provides information about the NBiometricsStandards library.

Files

Name	Description
NBiometricStandards.h (see page 74)	Header file of the NBiometricStandards (see page 13) library.

Functions

	Name	Description
	NBiometricStandardsGetInfo (see page 74)	Returns NLibraryInfo structure which contains information about the library.

6.1.1.6.1 Functions

6.1.1.6.1.1 NBiometricStandardsGetInfo Function

Returns NLibraryInfo structure which contains information about the library.

C++

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

Parameters

Parameters	Description
NLibraryInfo * pValue	[out] Pointer to NLibraryInfo object.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	is NULL (see page 133).

Module

NBiometricStandards Module ([see page 73](#))

6.1.1.6.2 Files

6.1.1.6.2.1 NBiometricStandards.h

Header file of the NBiometricStandards ([see page 13](#)) library.

Functions

	Name	Description
	NBiometricStandardsGetInfo (see page 74)	Returns NLibraryInfo structure which contains information about the library.

Module

NBiometricStandards Module ([see page 73](#))

6.1.2 NCore Library

Provides infrastructure for Neurotechnology components.

Remarks**Requirements (Windows)**

- **Import Library:** NCore.dll.lib.
- **DLL:** NCore.dll.

Requirements (Linux)

- **Shared object:** libNCore.so.

Modules

Name	Description
NCore Module (see page 75)	Provides infrastructure/basic functionality for Neurotechnology components.
NErrors Module (see page 77)	Defines error codes used in Neurotechnology components.
NGeometry Module (see page 84)	Provides definitions of geometrical structures types.
NLibraryInfo Module (see page 88)	Provides definitions of library info structure type.
NMemory Module (see page 91)	Provides memory management for Neurotechnology components.
NObject Module (see page 96)	Provides functionality for retrieving information about specified object.
NParameters Module (see page 101)	Provides functionality for working with parameters for Neurotechnology components.
NProcessorInfo Module (see page 106)	Provides functionality for getting processor information.
NTypes Module (see page 112)	Defines types and macros used in Neurotechnology components.

6.1.2.1 NCore Module

Provides infrastructure/basic functionality for Neurotechnology components.

Files

Name	Description
NCore.h (see page 77)	Header file for the NCore (see page 74) module. Provides infrastructure/basic functionality for Neurotechnology components.

Functions

	Name	Description
≡	NCoreGetInfo (see page 75)	Retrieves information about the library.
≡	NCoreOnExit (see page 76)	This function should be called on programs exit.
≡	NCoreOnStart (see page 76)	This function should be called on programs start.
≡	NCoreOnThreadExit (see page 76)	This function should be called when program's thread finished its task.
≡	NCoreOnThreadStart (see page 76)	This function should be called when new thread is created.

6.1.2.1.1 Functions

6.1.2.1.1.1 NCoreGetInfo Function

Retrieves information about the library.

C++

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

Parameters

Parameters	Description
NLibraryInfo * pValue	[out] Pointer to NLibraryInfo structure that receives library information.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NCore Module (see page 75)

6.1.2.1.1.2 NCoreOnExit Function

This function should be called on programs exit.

C++

```
void N_API NCoreOnExit();
```

Notes

This function works on Lib-No-Dll and VAR products.

Module

NCore Module (see page 75)

6.1.2.1.1.3 NCoreOnStart Function

This function should be called on programs start.

C++

```
void N_API NCoreOnStart();
```

Notes

This function works on Lib-No-Dll and VAR products.

Module

NCore Module (see page 75)

6.1.2.1.1.4 NCoreOnThreadExit Function

This function should be called when program's thread finished its task.

C++

```
void N_API NCoreOnThreadExit();
```

Notes

This function works on Lib-No-Dll and VAR products.

Module

NCore Module (see page 75)

6.1.2.1.1.5 NCoreOnThreadStart Function

This function should be called when new thread is created.

C++

```
void N_API NCoreOnThreadStart();
```

Notes

This function works on Lib-No-Dll and VAR products.

Module

NCore Module ([see page 75](#))

6.1.2.1.2 Files**6.1.2.1.2.1 NCore.h**

Header file for the NCore ([see page 74](#)) module. Provides infrastructure/basic functionality for Neurotechnology components.

Functions

	Name	Description
≡	NCoreGetInfo (see page 75)	Retrieves information about the library.
≡	NCoreOnExit (see page 76)	This function should be called on programs exit.
≡	NCoreOnStart (see page 76)	This function should be called on programs start.
≡	NCoreOnThreadExit (see page 76)	This function should be called when program's thread finished its task.
≡	NCoreOnThreadStart (see page 76)	This function should be called when new thread is created.

Module

NCore Module ([see page 75](#))

6.1.2.2 NErrors Module

Defines error codes used in Neurotechnology components.

Files

Name	Description
NErrors.h (see page 83)	Header file for NErrors module. Defines error codes used in Neurotechnology components.

Macros

Name	Description
N_E_ARGUMENT (see page 78)	This error occurs when argument value is invalid.
N_E_ARGUMENT_NULL (see page 78)	When argument value is NULL (see page 133) where non-NULL (see page 133) value was expected this error occurs.
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	Argument value is out of range.
N_E_CLR (see page 79)	The Common language runtime (CLR) exception has occurred.
N_E_COM (see page 79)	COM error has occurred.
N_E_CORE (see page 79)	Standard error has occurred (for internal use).
N_E_END_OF_STREAM (see page 79)	This error occurs when is attempted to read file or buffer after its end.
N_E_EXTERNAL (see page 79)	Error in external code has occurred (for internal use).
N_E_FAILED (see page 79)	Unspecified error has occurred.
N_E_FORMAT (see page 80)	This error occurs when format of argument value is invalid.
N_E_INDEX_OUT_OF_RANGE (see page 80)	This error occurs when index is out of range (for internal use).

N_E_INVALID_OPERATION (see page 80)	Attempted to perform invalid operation.
N_E_IO (see page 80)	Input/output error has occurred.
N_E_NOT_ACTIVATED (see page 80)	Product not activated.
N_E_NOT_IMPLEMENTED (see page 81)	This error occurs when trying to use functionality which is not implemented yet.
N_E_NOT_SUPPORTED (see page 81)	This error occurs when trying to use functionality which is not supported.
N_E_NULL_REFERENCE (see page 81)	Null reference has occurred (for internal use).
N_E_OUT_OF_MEMORY (see page 81)	This error occurs when there were not enough memory to proceed task.
N_E_OVERFLOW (see page 81)	Arithmetic overflow has occurred.
N_E_PARAMETER (see page 81)	Parameter id is invalid.
N_E_PARAMETER_READ_ONLY (see page 82)	Attempted to set read only parameter.
N_E_SYS (see page 82)	System error.
N_E_WIN32 (see page 82)	Win32 error has occurred.
N_OK (see page 82)	This value is returned when no error has occurred.
NFailed (see page 82)	!N_NO_ERROR_CODES Determines whether function result indicates error.
NSucceeded (see page 83)	Determines whether function result indicates success.

6.1.2.2.1 Macros

6.1.2.2.1.1 N_E_ARGUMENT Macro

This error occurs when argument value is invalid.

C++

```
#define N_E_ARGUMENT -10
```

Module

NErrors Module (see page 77)

6.1.2.2.1.2 N_E_ARGUMENT_NULL Macro

When argument value is NULL (see page 133) where non-NULL (see page 133) value was expected this error occurs.

C++

```
#define N_E_ARGUMENT_NULL -11
```

Module

NErrors Module (see page 77)

6.1.2.2.1.3 N_E_ARGUMENT_OUT_OF_RANGE Macro

Argument value is out of range.

C++

```
#define N_E_ARGUMENT_OUT_OF_RANGE -12
```

Module

NErrors Module (see page 77)

6.1.2.2.1.4 N_E_CLR Macro

The Common language runtime (CLR) exception has occurred.

C++

```
#define N_E_CLR -93
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.5 N_E_COM Macro

COM error has occurred.

C++

```
#define N_E_COM -92
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.6 N_E_CORE Macro

Standard error has occurred (for internal use).

C++

```
#define N_E_CORE -2
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.7 N_E_END_OF_STREAM Macro

This error occurs when is attempted to read file or buffer after its end.

C++

```
#define N_E_END_OF_STREAM -15
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.8 N_E_EXTERNAL Macro

Error in external code has occurred (for internal use).

C++

```
#define N_E_EXTERNAL -90
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.9 N_E_FAILED Macro

Unspecified error has occurred.

C++

```
#define N_E_FAILED -1
```


Module

NErrors Module ([see page 77](#))

6.1.2.2.1.10 N_E_FORMAT Macro

This error occurs when format of argument value is invalid.

C++

```
#define N_E_FORMAT -13
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.11 N_E_INDEX_OUT_OF_RANGE Macro

This error occurs when index is out of range (for internal use).

C++

```
#define N_E_INDEX_OUT_OF_RANGE -9
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.12 N_E_INVALID_OPERATION Macro

Attempted to perform invalid operation.

C++

```
#define N_E_INVALID_OPERATION -7
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.13 N_E_IO Macro

Input/output error has occurred.

C++

```
#define N_E_IO -14
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.14 N_E_NOT_ACTIVATED Macro

Product not activated.

C++

```
#define N_E_NOT_ACTIVATED -200
```

See Also

For more information about product activation see [Activation.pdf](#) within documentation folder of SDK.

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.15 N_E_NOT_IMPLEMENTED Macro

This error occurs when trying to use functionality which is not implemented yet.

C++

```
#define N_E_NOT_IMPLEMENTED -5
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.16 N_E_NOT_SUPPORTED Macro

This error occurs when trying to use functionality which is not supported.

C++

```
#define N_E_NOT_SUPPORTED -6
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.17 N_E_NULL_REFERENCE Macro

Null reference has occurred (for internal use).

C++

```
#define N_E_NULL_REFERENCE -3
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.18 N_E_OUT_OF_MEMORY Macro

This error occurs when there were not enough memory to proceed task.

C++

```
#define N_E_OUT_OF_MEMORY -4
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.19 N_E_OVERFLOW Macro

Arithmetic overflow has occurred.

C++

```
#define N_E_OVERFLOW -8
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.1.20 N_E_PARAMETER Macro

Parameter id is invalid.

C++

```
#define N_E_PARAMETER -100
```

Module

NErrors Module (see page 77)

6.1.2.2.1.21 N_E_PARAMETER_READ_ONLY Macro

Attempted to set read only parameter.

C++

```
#define N_E_PARAMETER_READ_ONLY -101
```

Module

NErrors Module (see page 77)

6.1.2.2.1.22 N_E_SYS Macro

System error.

C++

```
#define N_E_SYS -94
```

Module

NErrors Module (see page 77)

6.1.2.2.1.23 N_E_WIN32 Macro

Win32 error has occurred.

C++

```
#define N_E_WIN32 -91
```

Module

NErrors Module (see page 77)

6.1.2.2.1.24 N_OK Macro

This value is returned when no error has occurred.

C++

```
#define N_OK 0
```

Module

NErrors Module (see page 77)

6.1.2.2.1.25 NFailed Macro

!N_NO_ERROR_CODES

Determines whether function result indicates error.

C++

```
#define NFailed(result) ((result) < 0)
```

Module

NErrors Module (see page 77)

6.1.2.2.1.26 NSucceeded Macro

Determines whether function result indicates success.

C++

```
#define NSucceeded(result) ((result) >= 0)
```

Module

NErrors Module ([see page 77](#))

6.1.2.2.2 Files

6.1.2.2.2.1 NErrors.h

Header file for NErrors module. Defines error codes used in Neurotechnology components.

Macros

Name	Description
N_E_ARGUMENT (see page 78)	This error occurs when argument value is invalid.
N_E_ARGUMENT_NULL (see page 78)	When argument value is NULL (see page 133) where non-NULL (see page 133) value was expected this error occurs.
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	Argument value is out of range.
N_E_CLR (see page 79)	The Common language runtime (CLR) exception has occurred.
N_E_COM (see page 79)	COM error has occurred.
N_E_CORE (see page 79)	Standard error has occurred (for internal use).
N_E_END_OF_STREAM (see page 79)	This error occurs when is attempted to read file or buffer after its end.
N_E_EXTERNAL (see page 79)	Error in external code has occurred (for internal use).
N_E_FAILED (see page 79)	Unspecified error has occurred.
N_E_FORMAT (see page 80)	This error occurs when format of argument value is invalid.
N_E_INDEX_OUT_OF_RANGE (see page 80)	This error occurs when index is out of range (for internal use).
N_E_INVALID_OPERATION (see page 80)	Attempted to perform invalid operation.
N_E_IO (see page 80)	Input/output error has occurred.
N_E_NOT_ACTIVATED (see page 80)	Product not activated.
N_E_NOT_IMPLEMENTED (see page 81)	This error occurs when trying to use functionality which is not implemented yet.
N_E_NOT_SUPPORTED (see page 81)	This error occurs when trying to use functionality which is not supported.
N_E_NULL_REFERENCE (see page 81)	Null reference has occurred (for internal use).
N_E_OUT_OF_MEMORY (see page 81)	This error occurs when there were not enough memory to proceed task.
N_E_OVERFLOW (see page 81)	Arithmetic overflow has occurred.
N_E_PARAMETER (see page 81)	Parameter id is invalid.
N_E_PARAMETER_READ_ONLY (see page 82)	Attempted to set read only parameter.
N_E_SYS (see page 82)	System error.
N_E_WIN32 (see page 82)	Win32 error has occurred.
N_OK (see page 82)	This value is returned when no error has occurred.
NFailed (see page 82)	!N_NO_ERROR_CODES Determines whether function result indicates error.
NSucceeded (see page 83)	Determines whether function result indicates success.

Module

NErrors Module ([see page 77](#))

6.1.2.3 NGeometry Module

Provides definitions of geometrical structures types.

Files

Name	Description
NGeometry.h (see page 87)	Header file for the NGeometry module. Provides definitions of geometrical structures types.

Structs, Records, Enums

	Name	Description
	NPoint (see page 84)	Structure defining point coordinates in 2D space.
	NRect (see page 85)	Structure defining a rectangle figure in 2D space.
	NSize (see page 85)	Structure defining rectangle size.
	NRect_ (see page 85)	Structure defining a rectangle figure in 2D space.
	NPoint_ (see page 84)	Structure defining point coordinates in 2D space.
	NSize_ (see page 85)	Structure defining rectangle size.
	NPointD_ (see page 85)	Structure defining point coordinates in 2D space.
	NPointF_ (see page 86)	Structure defining point coordinates in 2D space.
	NRectD_ (see page 86)	Structure defining a rectangle figure in 2D space.
	NRectF_ (see page 86)	Structure defining a rectangle figure in 2D space.
	NSizeD_ (see page 87)	Structure defining rectangle size.
	NSizeF_ (see page 87)	Structure defining rectangle size.
	NPointD (see page 85)	Structure defining point coordinates in 2D space.
	NPointF (see page 86)	Structure defining point coordinates in 2D space.
	NRectD (see page 86)	Structure defining a rectangle figure in 2D space.
	NRectF (see page 86)	Structure defining a rectangle figure in 2D space.
	NSizeD (see page 87)	Structure defining rectangle size.
	NSizeF (see page 87)	Structure defining rectangle size.

6.1.2.3.1 Structs, Records, Enums

6.1.2.3.1.1 NPoint Structure

Structure defining point coordinates in 2D space.

C++

```
typedef struct NPoint_ {
    NInt X;
    NInt Y;
} NPoint;
```

Members

Members	Description
NInt X;	Point coordinate on x axis.
NInt Y;	Point coordinate on y axis.

ModuleNGeometry Module ([see page 84](#))**6.1.2.3.1.2 NRect Structure**

Structure defining a rectangle figure in 2D space.

C++

```
typedef struct NRect_ {  
    NInt X;  
    NInt Y;  
    NInt Width;  
    NInt Height;  
} NRect;
```

Members

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

ModuleNGeometry Module ([see page 84](#))**6.1.2.3.1.3 NSize Structure**

Structure defining rectangle size.

C++

```
typedef struct NSize_ {  
    NInt Width;  
    NInt Height;  
} NSize;
```

Members

Members	Description
NInt Width;	Width.
NInt Height;	Height.

ModuleNGeometry Module ([see page 84](#))**6.1.2.3.1.4 NPointD Structure**

Structure defining point coordinates in 2D space.

C++

```
typedef struct NPointD_ {  
    NDouble X;  
    NDouble Y;  
} NPointD;
```

Members

Members	Description
NDouble X;	Point coordinate on x axis.

NDouble Y;	Point coordinate on y axis.

Module

NGeometry Module ([see page 84](#))

6.1.2.3.1.5 NPointF Structure

Structure defining point coordinates in 2D space.

C++

```
typedef struct NPointF_ {  
    NFloat X;  
    NFloat Y;  
} NPointF;
```

Members

Members	Description
NFloat X;	Point coordinate on x axis.
NFloat Y;	Point coordinate on y axis.

Module

NGeometry Module ([see page 84](#))

6.1.2.3.1.6 NRectD Structure

Structure defining a rectangle figure in 2D space.

C++

```
typedef struct NRectD_ {  
    NDouble X;  
    NDouble Y;  
    NDouble Width;  
    NDouble Height;  
} NRectD;
```

Members

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

Module

NGeometry Module ([see page 84](#))

6.1.2.3.1.7 NRectF Structure

Structure defining a rectangle figure in 2D space.

C++

```
typedef struct NRectF_ {  
    NFloat X;  
    NFloat Y;  
    NFloat Width;  
    NFloat Height;  
} NRectF;
```

Members

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

Module

NGeometry Module ([↗](#) see page 84)

6.1.2.3.1.8 NSized Structure

Structure defining rectangle size.

C++

```
typedef struct NSized_ {  
    NDouble width;  
    NDouble Height;  
} NSized;
```

Members

Members	Description
NDouble Width;	Width.
NDouble Height;	Height.

Module

NGeometry Module ([↗](#) see page 84)

6.1.2.3.1.9 NSizeF Structure

Structure defining rectangle size.

C++

```
typedef struct NSizeF_ {  
    NFloat width;  
    NFloat Height;  
} NSizeF;
```

Members

Members	Description
NFloat Width;	Width.
NFloat Height;	Height.

Module

NGeometry Module ([↗](#) see page 84)

6.1.2.3.2 Files

6.1.2.3.2.1 NGeometry.h

Header file for the NGeometry module. Provides definitions of geometrical structures types.

Module

NGeometry Module ([↗](#) see page 84)

Structures

	Name	Description
	NPoint_ (see page 84)	Structure defining point coordinates in 2D space.
	NPointD_ (see page 85)	Structure defining point coordinates in 2D space.
	NPointF_ (see page 86)	Structure defining point coordinates in 2D space.
	NRect_ (see page 85)	Structure defining a rectangle figure in 2D space.
	NRectD_ (see page 86)	Structure defining a rectangle figure in 2D space.
	NRectF_ (see page 86)	Structure defining a rectangle figure in 2D space.
	NSize_ (see page 85)	Structure defining rectangle size.
	NSizeD_ (see page 87)	Structure defining rectangle size.
	NSizeF_ (see page 87)	Structure defining rectangle size.
	NPoint (see page 84)	Structure defining point coordinates in 2D space.
	NPointD (see page 85)	Structure defining point coordinates in 2D space.
	NPointF (see page 86)	Structure defining point coordinates in 2D space.
	NRect (see page 85)	Structure defining a rectangle figure in 2D space.
	NRectD (see page 86)	Structure defining a rectangle figure in 2D space.
	NRectF (see page 86)	Structure defining a rectangle figure in 2D space.
	NSize (see page 85)	Structure defining rectangle size.
	NSizeD (see page 87)	Structure defining rectangle size.
	NSizeF (see page 87)	Structure defining rectangle size.

6.1.2.4 NLibraryInfo Module

Provides definitions of library info structure type.

Files

Name	Description
NLibraryInfo.h (see page 90)	Header file for the NLibraryInfo module. Provides definitions of library info structure type.

Macros

Name	Description
N_LI_ACTIVATED_MAX_LENGTH (see page 89)	Maximum length of "Activated" field.
N_LI_COMPANY_MAX_LENGTH (see page 89)	Maximum length of company name.
N_LI_COPYRIGHT_MAX_LENGTH (see page 90)	Maximum length of copyright notice.
N_LI_PRODUCT_MAX_LENGTH (see page 90)	Maximum length of product name.
N_LI_TITLE_MAX_LENGTH (see page 90)	Maximum length of library title.

Structs, Records, Enums

	Name	Description
	NLibraryInfo_ (see page 89)	Structure defining information about the library as library title, product name, company name, copyright string and library version.

6.1.2.4.1 Structs, Records, Enums

6.1.2.4.1.1 NLibraryInfo_ Structure

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

C++

```
struct NLibraryInfo_ {
    NChar Title[N_LI_TITLE_MAX_LENGTH];
    NChar Product[N_LI_PRODUCT_MAX_LENGTH];
    NChar Company[N_LI_COMPANY_MAX_LENGTH];
    NChar Copyright[N_LI_COPYRIGHT_MAX_LENGTH];
    NInt VersionMajor;
    NInt VersionMinor;
    NInt VersionBuild;
    NInt VersionRevision;
    NChar Activated[N_LI_ACTIVATED_MAX_LENGTH];
};
```

Members

Members	Description
NChar Title[N_LI_TITLE_MAX_LENGTH];	Title of the library.
NChar Product[N_LI_PRODUCT_MAX_LENGTH];	Product name.
NChar Company[N_LI_COMPANY_MAX_LENGTH];	Company name
NChar Copyright[N_LI_COPYRIGHT_MAX_LENGTH];	Copyright string for the library.
NInt VersionMajor;	Major part of the library version.
NInt VersionMinor;	Minor part of the library version.
NInt VersionBuild;	Build part of the library version.
NInt VersionRevision;	Revision part of the library version.
NChar Activated[N_LI_ACTIVATED_MAX_LENGTH];	Indicates if the library was activated.

Module

NLibraryInfo Module (see page 88)

6.1.2.4.2 Macros

6.1.2.4.2.1 N_LI_ACTIVATED_MAX_LENGTH Macro

Maximum length of "Activated" field.

C++

```
#define N_LI_ACTIVATED_MAX_LENGTH 256
```

Module

NLibraryInfo Module (see page 88)

6.1.2.4.2.2 N_LI_COMPANY_MAX_LENGTH Macro

Maximum length of company name.

C++

```
#define N_LI_COMPANY_MAX_LENGTH 64
```

Module

NLibraryInfo Module (see page 88)

6.1.2.4.2.3 N_LI_COPYRIGHT_MAX_LENGTH Macro

Maximum length of copyright notice.

C++

```
#define N_LI_COPYRIGHT_MAX_LENGTH 192
```

Module

NLibraryInfo Module (see page 88)

6.1.2.4.2.4 N_LI_PRODUCT_MAX_LENGTH Macro

Maximum length of product name.

C++

```
#define N_LI_PRODUCT_MAX_LENGTH 128
```

Module

NLibraryInfo Module (see page 88)

6.1.2.4.2.5 N_LI_TITLE_MAX_LENGTH Macro

Maximum length of library title.

C++

```
#define N_LI_TITLE_MAX_LENGTH 128
```

Module

NLibraryInfo Module (see page 88)

6.1.2.4.3 Files

6.1.2.4.3.1 NLibraryInfo.h

Header file for the NLibraryInfo module. Provides definitions of library info structure type.

Macros

Name	Description
N_LI_ACTIVATED_MAX_LENGTH (see page 89)	Maximum length of "Activated" field.
N_LI_COMPANY_MAX_LENGTH (see page 89)	Maximum length of company name.
N_LI_COPYRIGHT_MAX_LENGTH (see page 90)	Maximum length of copyright notice.
N_LI_PRODUCT_MAX_LENGTH (see page 90)	Maximum length of product name.
N_LI_TITLE_MAX_LENGTH (see page 90)	Maximum length of library title.

Module

NLibraryInfo Module (see page 88)

Structures

	Name	Description
	NLibraryInfo_ (see page 89)	Structure defining information about the library as library title, product name, company name, copyright string and library version.

6.1.2.5 NMemory Module

Provides memory management for Neurotechnology components.

Files

Name	Description
NMemory.h (see page 95)	Header file for the NMemory module. Provides memory management for Neurotechnology components.

Functions

	Name	Description
◆	NAlignedFree (see page 91)	Frees an aligned memory resources.
◆	NAlloc (see page 91)	Allocates memory block.
◆	NCAAlloc (see page 92)	Allocates memory block with all bytes set to zero.
◆	NCompare (see page 92)	Compares bytes in two memory blocks.
◆	NCopy (see page 93)	Copies data between memory blocks.
◆	NFill (see page 93)	Sets bytes of memory block to specified value.
◆	NFree (see page 94)	Deallocates memory block.
◆	NMove (see page 94)	Move data from one memory block to another.
◆	NReAlloc (see page 95)	Reallocates memory block.

Macros

Name	Description
NClear (see page 95)	Clears memory block.

6.1.2.5.1 Functions

6.1.2.5.1.1 NAlignedFree Function

Frees an aligned memory resources.

C++

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

Parameters

Parameters	Description
void * pBlock	[out] A pointer to previously allocated memory block to free.

Remarks

If passed NULL (see page 133) as a parameter, function does nothing. Attempting to free an invalid pointer, can cause errors.

Module

NMemory Module (see page 91)

6.1.2.5.1.2 NAlloc Function

Allocates memory block.

C++

```
NResult N_API NAlloc(NSizeType size, void * * ppBlock);
```

Parameters

Parameters	Description
NSizeType size	The size in bytes of memory to be allocated.
void ** ppBlock	[out] A pointer to a newly allocated memory block.

Returns

Return value	Description
N_OK (see page 82)	If function succeeds it returns N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If provided <i>ppBlock</i> value is NULL (see page 133), function fails and returns this error.
N_E_OUT_OF_MEMORY (see page 81)	If there was not enough memory to allocate <i>size</i> bytes this error is returned.

Module

NMemory Module (see page 91)

6.1.2.5.1.3 NAlloc Function

Allocates memory block with all bytes set to zero.

C++

```
NResult N_API NAlloc(NSizeType size, void ** ppBlock);
```

Parameters

Parameters	Description
NSizeType size	The size in bytes of memory to be allocated.
void ** ppBlock	[out] A pointer to a newly allocated memory block.

Returns

Return value	Description
N_OK (see page 82)	If function succeeds it returns N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If provided <i>ppBlock</i> value is NULL (see page 133), function fails and returns this error.
N_E_OUT_OF_MEMORY (see page 81)	If there was not enough memory to allocate <i>size</i> bytes this error is returned.

Module

NMemory Module (see page 91)

6.1.2.5.1.4 NCompare Function

Compares bytes in two memory blocks.

C++

```
NResult N_API NCompare(const void * pBlock1, const void * pBlock2, NSizeType size, NInt * pResult);
```

Parameters

Parameters	Description
const void * pBlock1	A pointer to the first memory block.
const void * pBlock2	A pointer to the second memory block.

NSizeType size	Number of bytes to be compared.
NInt * pResult	[out] Compare result.

Returns

Return value	Description
N_OK (see page 82)	If function succeeds it returns N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If provided <i>ppBlock1</i> or <i>ppBlock2</i> value is NULL (see page 133), function fails and returns this error.

Remarks

If *pResult* value is less than zero then a block specified by *pBlock1* is less than a block specified by *pBlock2*.

If *pResult* has a zero value then a block specified by *pBlock1* is identical to a block specified by *pBlock2*.

If *pResult* is greater than zero then A block specified by *pBlock1* is greater than a block specified by *pBlock2*.

Module

NMemory Module (see page 91)

6.1.2.5.1.5 NCopy Function

Copies data between memory blocks.

C++

```
NResult N_API NCopy(void * pDstBlock, const void * pSrcBlock, NSizeType size);
```

Parameters

Parameters	Description
void * pDstBlock	A destination memory block (memory block to copy from).
const void * pSrcBlock	A source memory block (memory block to copy to).
NSizeType size	The size of memory block to be copied (the size is in bytes).

Returns

Return value	Description
N_OK (see page 82)	If function succeeds it returns N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If provided <i>pDstBlock</i> or <i>pSrcBlock</i> value is NULL (see page 133), function fails and returns this error.

Module

NMemory Module (see page 91)

6.1.2.5.1.6 NFill Function

Sets bytes of memory block to specified value.

C++

```
NResult N_API NFill(void * pBlock, NByte value, NSizeType size);
```

Parameters

Parameters	Description
void * pBlock	A pointer to a destination memory block that contains data.

NByte value	A value to set.
NSizeType size	The size in bytes of memory block to be set.

Returns

Return value	Description
N_OK (see page 82)	If function succeeds it returns N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If provided <i>pBlock</i> value is NULL (see page 133), function fails and returns this error.

Module

NMemory Module (see page 91)

6.1.2.5.1.7 NFree Function

Deallocates memory block.

C++

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

Parameters

Parameters	Description
void * pBlock	A pointer to memory block to be deallocated.

Returns

If function succeeds the return value is N_OK (see page 82).

Module

NMemory Module (see page 91)

6.1.2.5.1.8 NMove Function

Move data from one memory block to another.

C++

```
NResult N_API NMove(void * pDstBlock, const void * pSrcBlock, NSizeType size);
```

Parameters

Parameters	Description
void * pDstBlock	A pointer to destination memory block.
const void * pSrcBlock	A pointer to source memory block.
NSizeType size	Size in bytes to be copied.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Remarks

Copies *size* bytes from *pDstBlock* to *pSrcBlock*.

Make sure that the destination buffer is the same size or larger than the source buffer.

Module

NMemory Module (see page 91)

6.1.2.5.1.9 NReAlloc Function

Reallocates memory block.

C++

```
NResult N_API NReAlloc(void * * ppBlock, NSizeType size);
```

Parameters

Parameters	Description
void * * ppBlock	Pointer to previously allocated memory block which should be reallocated.
NSizeType size	The size in bytes of a new memory block.

Returns

Return value	Description
N_OK (see page 82)	If function succeeds it returns N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If provided <i>ppBlock</i> value is NULL (see page 133), function fails and returns this error.
N_E_OUT_OF_MEMORY (see page 81)	If there was not enough memory to allocate <i>size</i> bytes this error is returned.

Module

NMemory Module (see page 91)

6.1.2.5.2 Macros

6.1.2.5.2.1 NClear Macro

Clears memory block.

C++

```
#define NClear(pBlock, size) NFill(pBlock, 0, size)
```

Module

NMemory Module (see page 91)

6.1.2.5.3 Files

6.1.2.5.3.1 NMemory.h

Header file for the NMemory module. Provides memory management for Neurotechnology components.

Functions

	Name	Description
⇒	NAlignedFree (see page 91)	Frees an aligned memory resources.
⇒	NAlloc (see page 91)	Allocates memory block.
⇒	NCAAlloc (see page 92)	Allocates memory block with all bytes set to zero.
⇒	NCompare (see page 92)	Compares bytes in two memory blocks.
⇒	NCopy (see page 93)	Copies data between memory blocks.
⇒	NFill (see page 93)	Sets bytes of memory block to specified value.
⇒	NFree (see page 94)	Deallocates memory block.

	NMove (see page 94)	Move data from one memory block to another.
	NReAlloc (see page 95)	Reallocates memory block.

Macros

Name	Description
NClear (see page 95)	Clears memory block.

Module

NMemory Module ([see page 91](#))

6.1.2.6 NObject Module

Provides functionality for retrieving information about specified object.

Files

Name	Description
NObject.h (see page 101)	Header file for NObject module. Provides functionality for retrieving information about specified object.

Functions

	Name	Description
	NObjectCopyParameters (see page 96)	Copies parameter values from one object to another.
	NObjectFree (see page 97)	Deletes the object specified by handle. After the object is deleted the specified handle is no longer valid.
	NObjectGetOwner (see page 97)	Gets owner of specified object.
	NObjectGetParameter (see page 97)	Retrieves value of the parameter specified by Id.
	NObjectGetParameterWithPart (see page 98)	Retrieves value of the parameter specified by Id.
	NObjectGetType (see page 98)	Retrieves type of specified object.
	NObjectReset (see page 99)	Sets default values for all parameters of the specified object.
	NObjectSetParameter (see page 99)	Sets value of the specified parameter of the specified object.
	NObjectSetParameterWithPart (see page 99)	Sets value of the specified parameter of the specified object.
	NObjectTypeOf (see page 100)	Retrieves object type.
	NTypeGetBaseType (see page 100)	Retrieves base type of specified type.
	NTypeGetName (see page 101)	Retrieves name of specified type.

6.1.2.6.1 Functions

6.1.2.6.1.1 NObjectCopyParameters Function

Copies parameter values from one object to another.

C++

```
NResult N_API NObjectCopyParameters(HObject hDstObject, HObject hSrcObject);
```

Parameters

Parameters	Description
HNObject hDstObject	[in] Destination object. Source object will be copied to this object.
HNObject hSrcObject	[in] Source object to copy.

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned.

Module

NObject Module (see page 96)

6.1.2.6.1.2 NObjectFree Function

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

C++

```
void N_API NObjectFree(HNObject hObject);
```

Parameters

Parameters	Description
HNObject hObject	[in] Handle to object to be deleted.

Module

NObject Module (see page 96)

6.1.2.6.1.3 NObjectGetOwner Function

Gets owner of specified object.

C++

```
NResult N_API NObjectGetOwner(HNObject hObject, HNObject * pValue);
```

Parameters

Parameters	Description
HNObject hObject	[in] Handle to object to retrieve it's owner.
HNObject * pValue	[out] Pointer to owner object.

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned.

Module

NObject Module (see page 96)

6.1.2.6.1.4 NObjectGetParameter Function

Retrieves value of the parameter specified by Id.

C++

```
NResult N_API NObjectGetParameter(HNObject hObject, NUInt parameterId, void * pValue);
```

Parameters

Parameters	Description
HNObject hObject	[in] Handle to object to retrieve it's parameter value.
NUInt parameterId	[in] Identifier of the parameter to retrieve.

<code>void * pValue</code>	[out] Pointer to variable that receives parameter value.

Returns

If the function succeeds and *parameterId* specifies a `N_TYPE_STRING` (see page 104) type parameter, and *pValue* is `NULL` (see page 133), the return value is length of the string (not including the `NULL` (see page 133)-terminator) *pValue* should point to. If the function succeeds and *pValue* is not `NULL` (see page 133), the return value is `N_OK` (see page 82). Otherwise, one of the following error codes is returned.

Remarks

To learn the type of the parameter pass value obtained with `NParameterMakeld` (see page 105) macro using `N_PC_TYPE_ID` (see page 102) code and the parameter id via *parameterId* parameter and pointer to `NInt` (see page 117) that will receive one of `N_TYPE_XXX` via *pValue* parameter. *hObject* can be `NULL` (see page 133) in this case.

Module

NObject Module (see page 96)

6.1.2.6.1.5 NObjectGetParameterWithPart Function

Retrieves value of the parameter specified by Id.

C++

```
NResult N_API NObjectGetParameterWithPart(HNObject hObject, NUShort partId, NUInt parameterId, void * pValue);
```

Parameters

Parameters	Description
<code>HNObject hObject</code>	[in] Handle to object to retrieve it's parameter value.
<code>NUShort partId</code>	[in] Parameter part Id.
<code>NUInt parameterId</code>	[in] Identifier of the parameter to retrieve.
<code>void * pValue</code>	[out] Pointer to variable that receives parameter value.

Returns

If the function succeeds and *parameterId* specifies a `N_TYPE_STRING` (see page 104) type parameter, and *pValue* is `NULL` (see page 133), the return value is length of the string (not including the `NULL` (see page 133)-terminator) *pValue* should point to. If the function succeeds and *pValue* is not `NULL` (see page 133), the return value is `N_OK` (see page 82). Otherwise, one of the following error codes is returned.

Remarks

To learn the type of the parameter pass value obtained with `NParameterMakeld` (see page 105) macro using `N_PC_TYPE_ID` (see page 102) code and the parameter id via *parameterId* parameter and pointer to `NInt` (see page 117) that will receive one of `N_TYPE_XXX` via *pValue* parameter. *hObject* can be `NULL` (see page 133) in this case.

Module

NObject Module (see page 96)

6.1.2.6.1.6 NObjectGetType Function

Retrieves type of specified object.

C++

```
NResult N_API NObjectGetType(HNObject hObject, HNTYPE * pValue);
```

Parameters

Parameters	Description
<code>HNObject hObject</code>	[in] Handle to object to retrieve it's type.

HNType * pValue	[out] Object's type value.

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned.

Module

NObject Module (see page 96)

6.1.2.6.1.7 NObjectReset Function

Sets default values for all parameters of the specified object.

C++

```
NResult N_API NObjectReset(HNObject hObject);
```

Parameters

Parameters	Description
HNObject hObject	[in] Handle to object to reset values.

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned.

Module

NObject Module (see page 96)

6.1.2.6.1.8 NObjectSetParameter Function

Sets value of the specified parameter of the specified object.

C++

```
NResult N_API NObjectSetParameter(HNObject hObject, NUInt parameterId, const void * pValue);
```

Parameters

Parameters	Description
HNObject hObject	[in] Handle to the object. Can be NULL (see page 133) if setting static parameter value.
NUInt parameterId	[in] Identifier of the parameter to set.
const void * pValue	[in] Pointer to the parameter value to set.

Returns

If the function succeeds and *parameterId* specifies a N_TYPE_STRING (see page 104) type parameter, and *pValue* is NULL (see page 133), the return value is length of the string (not including the NULL (see page 133)-terminator) *pValue* should point to. If the function succeeds and *pValue* is not NULL (see page 133), the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned.

Remarks

To learn the type of the parameter pass value obtained with NParameterMakeld (see page 105) macro using N_PC_TYPE_ID (see page 102) code and the parameter id via parameterId parameter and pointer to NInt (see page 117) that will receive one of N_TYPE_XXX via pValue parameter. hObject can be NULL (see page 133) in this case.

Module

NObject Module (see page 96)

6.1.2.6.1.9 NObjectSetParameterWithPart Function

Sets value of the specified parameter of the specified object.

C++

```
NResult N_API NObjectSetParameterWithPart(HNObject hObject, NUShort partId, NUInt  
parameterId, const void * pValue);
```

Parameters

Parameters	Description
HNObject hObject	[in] Handle to the object. Can be NULL (see page 133) if setting static parameter value.
NUShort partId	[in] Identifier of the part of the matcher.
NUInt parameterId	[in] Identifier of the parameter to set.
const void * pValue	[in] Pointer to the parameter value to set.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Remarks

To learn the type of the parameter pass value obtained with NParameterMakeId (see page 105) macro using N_PC_TYPE_ID (see page 102) code and the parameter id via parameterId parameter and pointer to NInt (see page 117) that will receive one of N_TYPE_XXX via pValue parameter. hObject can be NULL (see page 133) in this case.

Module

NObject Module (see page 96)

6.1.2.6.1.10 NObjectTypeOf Function

Retrieves object type.

C++

```
HNType N_API NObjectTypeOf();
```

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned.

Module

NObject Module (see page 96)

6.1.2.6.1.11 NTypeGetBaseType Function

Retrieves base type of specified type.

C++

```
NResult N_API NTypeGetBaseType(HNType hType, HNType * pValue);
```

Parameters

Parameters	Description
HNType hType	[in] Handle to type to retrieve its base type.
HNType * pValue	[out] Pointer to base type.

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned.

Module

NObject Module (see page 96)

6.1.2.6.1.12 NTypeGetName Function

Retrieves name of specified type.

C++

```
NResult N_API NTypeGetName(HNType hType, NChar * pValue);
```

Parameters

Parameters	Description
HNType hType	[in] Handle to type to retrieve it's type.
NChar * pValue	[out] Name of specified type.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)). Otherwise, one of the following error codes is returned.

Module

NObject Module ([see page 96](#))

6.1.2.6.2 Files

6.1.2.6.2.1 NObject.h

Header file for NObject ([see page 96](#)) module. Provides functionality for retrieving information about specified object.

Functions

	Name	Description
≡	NObjectCopyParameters (see page 96)	Copies parameter values from one object to another.
≡	NObjectFree (see page 97)	Deletes the object specified by handle. After the object is deleted the specified handle is no longer valid.
≡	NObjectGetOwner (see page 97)	Gets owner of specified object.
≡	NObjectGetParameter (see page 97)	Retrieves value of the parameter specified by Id.
≡	NObjectGetParameterWithPart (see page 98)	Retrieves value of the parameter specified by Id.
≡	NObjectGetType (see page 98)	Retrieves type of specified object.
≡	NObjectReset (see page 99)	Sets default values for all parameters of the specified object.
≡	NObjectSetParameter (see page 99)	Sets value of the specified parameter of the specified object.
≡	NObjectSetParameterWithPart (see page 99)	Sets value of the specified parameter of the specified object.
≡	NObjectTypeOf (see page 100)	Retrieves object type.
≡	NTypeGetBaseType (see page 100)	Retrieves base type of specified type.
≡	NTypeGetName (see page 101)	Retrieves name of specified type.

Module

NObject Module ([see page 96](#))

6.1.2.7 NParameters Module

Provides functionality for working with parameters for Neurotechnology components.

Files

Name	Description
NParameters.h (see page 105)	Header file for the NParameters module. Provides functionality for working with parameters for Neurotechnology components.

Macros

Name	Description
N_TYPE_BOOL (see page 102)	Specifies that parameter type is NBool (see page 116).
N_PC_TYPE_ID (see page 102)	Specifies that type id (NInt (see page 117) value, one of N_TYPE_XXX) of the parameter should be retrieved.
N_TYPE_BYTE (see page 102)	Specifies that parameter type is NByte (see page 116).
N_TYPE_CHAR (see page 103)	Specifies that parameter type is NChar (see page 117).
N_TYPE_DOUBLE (see page 103)	Specifies that parameter type is NDouble (see page 117).
N_TYPE_FLOAT (see page 103)	Specifies that parameter type is NFloat (see page 117).
N_TYPE_INT (see page 103)	Specifies that parameter type is NInt (see page 117).
N_TYPE_LONG (see page 103)	Specifies that parameter type is NLong (see page 118).
N_TYPE_SBYTE (see page 104)	Specifies that parameter type is NSByte (see page 119).
N_TYPE_SHORT (see page 104)	Specifies that parameter type is NShort (see page 119).
N_TYPE_STRING (see page 104)	Specifies that parameter type is null-terminated string of NChar (see page 117).
N_TYPE_UINT (see page 104)	Specifies that parameter type is NUInt (see page 119).
N_TYPE_ULONG (see page 104)	Specifies that parameter type is NULong (see page 120).
N_TYPE_USHORT (see page 104)	Specifies that parameter type is NUShort (see page 120).
NParameterMakeld (see page 105)	Makes parameter id.

6.1.2.7.1 Macros**6.1.2.7.1.1 N_TYPE_BOOL Macro**

Specifies that parameter type is NBool (see page 116).

C++

```
#define N_TYPE_BOOL 10
```

Module

NParameters Module (see page 101)

6.1.2.7.1.2 N_PC_TYPE_ID Macro

Specifies that type id (NInt (see page 117) value, one of N_TYPE_XXX) of the parameter should be retrieved.

C++

```
#define N_PC_TYPE_ID 1
```

Module

NParameters Module (see page 101)

6.1.2.7.1.3 N_TYPE_BYTE Macro

Specifies that parameter type is NByte (see page 116).

C++

```
#define N_TYPE_BYTE 1
```

ModuleNParameters Module ([↗](#) see page 101)

6.1.2.7.1.4 N_TYPE_CHAR Macro

Specifies that parameter type is NChar ([↗](#) see page 117).**C++**

```
#define N_TYPE_CHAR 20
```

ModuleNParameters Module ([↗](#) see page 101)

6.1.2.7.1.5 N_TYPE_DOUBLE Macro

Specifies that parameter type is NDouble ([↗](#) see page 117).**C++**

```
#define N_TYPE_DOUBLE 31
```

ModuleNParameters Module ([↗](#) see page 101)

6.1.2.7.1.6 N_TYPE_FLOAT Macro

Specifies that parameter type is NFloat ([↗](#) see page 117).**C++**

```
#define N_TYPE_FLOAT 30
```

ModuleNParameters Module ([↗](#) see page 101)

6.1.2.7.1.7 N_TYPE_INT Macro

Specifies that parameter type is NInt ([↗](#) see page 117).**C++**

```
#define N_TYPE_INT 6
```

ModuleNParameters Module ([↗](#) see page 101)

6.1.2.7.1.8 N_TYPE_LONG Macro

Specifies that parameter type is NLong ([↗](#) see page 118).**C++**

```
#define N_TYPE_LONG 8
```

ModuleNParameters Module ([↗](#) see page 101)

6.1.2.7.1.9 N_TYPE_SBYTE Macro

Specifies that parameter type is NSByte ([↗](#) see page 119).

C++

```
#define N_TYPE_SBYTE 2
```

Module

NParameters Module ([↗](#) see page 101)

6.1.2.7.1.10 N_TYPE_SHORT Macro

Specifies that parameter type is NShort ([↗](#) see page 119).

C++

```
#define N_TYPE_SHORT 4
```

Module

NParameters Module ([↗](#) see page 101)

6.1.2.7.1.11 N_TYPE_STRING Macro

Specifies that parameter type is null-terminated string of NChar ([↗](#) see page 117).

C++

```
#define N_TYPE_STRING 100
```

Module

NParameters Module ([↗](#) see page 101)

6.1.2.7.1.12 N_TYPE_UINT Macro

Specifies that parameter type is NUInt ([↗](#) see page 119).

C++

```
#define N_TYPE_UINT 5
```

Module

NParameters Module ([↗](#) see page 101)

6.1.2.7.1.13 N_TYPE_ULONG Macro

Specifies that parameter type is NULong ([↗](#) see page 120).

C++

```
#define N_TYPE_ULONG 7
```

Module

NParameters Module ([↗](#) see page 101)

6.1.2.7.1.14 N_TYPE_USHORT Macro

Specifies that parameter type is NUShort ([↗](#) see page 120).

C++

```
#define N_TYPE_USHORT 3
```

Module

NParameters Module ([see page 101](#))

6.1.2.7.1.15 NParameterMakeId Macro

Makes parameter id.

C++

```
#define NParameterMakeId(code, index, id) ((NUInt)((((code) << 24) | (((index) & 0xFF) << 16) | ((id) & 0xFFFF)))
```

Parameters

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

Module

NParameters Module ([see page 101](#))

6.1.2.7.2 Files**6.1.2.7.2.1 NParameters.h**

Header file for the NParameters module. Provides functionality for working with parameters for Neurotechnology components.

Macros

Name	Description
N_PC_TYPE_ID (see page 102)	Specifies that type id (NInt (see page 117) value, one of N_TYPE_XXX) of the parameter should be retrieved.
N_TYPE_BYTE (see page 102)	Specifies that parameter type is NByte (see page 116).
N_TYPE_BOOL (see page 102)	Specifies that parameter type is NBool (see page 116).
N_TYPE_CHAR (see page 103)	Specifies that parameter type is NChar (see page 117).
N_TYPE_DOUBLE (see page 103)	Specifies that parameter type is NDouble (see page 117).
N_TYPE_FLOAT (see page 103)	Specifies that parameter type is NFloat (see page 117).
N_TYPE_INT (see page 103)	Specifies that parameter type is NInt (see page 117).
N_TYPE_LONG (see page 103)	Specifies that parameter type is NLong (see page 118).
N_TYPE_SBYTE (see page 104)	Specifies that parameter type is NSByte (see page 119).
N_TYPE_SHORT (see page 104)	Specifies that parameter type is NShort (see page 119).
N_TYPE_STRING (see page 104)	Specifies that parameter type is null-terminated string of NChar (see page 117).
N_TYPE_UINT (see page 104)	Specifies that parameter type is NUInt (see page 119).
N_TYPE_ULONG (see page 104)	Specifies that parameter type is NULong (see page 120).
N_TYPE_USHORT (see page 104)	Specifies that parameter type is NUShort (see page 120).
NParameterMakeId (see page 105)	Makes parameter id.

Module

NParameters Module ([see page 101](#))

6.1.2.8 NProcessorInfo Module

Provides functionality for getting processor information.

Files

Name	Description
NProcessorInfo.h (see page 111)	Header file for the NProcessorInfo. Provides functionality for getting processor information.

Functions

	Name	Description
	NProcessorInfoGetModelNameA (see page 106)	Retrieves processor model name.
	NProcessorInfoGetModelNameW (see page 107)	Retrieves processor model name.
	NProcessorInfoGetVendor (see page 107)	Retrieves a pointer to enumeration that contains information about processor's vendor.
	NProcessorInfoGetVendorNameA (see page 108)	Retrieves processor's vendor name.
	NProcessorInfoGetVendorNameW (see page 108)	Retrieves processor's vendor name.
	NProcessorInfoIs3DNowSupported (see page 109)	
	NProcessorInfoIsMmxSupported (see page 109)	
	NProcessorInfoIsSse2Supported (see page 109)	
	NProcessorInfoIsSse3Supported (see page 110)	
	NProcessorInfoIsSseSupported (see page 110)	

Macros

Name	Description
N_PROCESSOR_INFO_MAX_MODEL_NAME_LENGTH (see page 111)	Maximum length of processor model.
N_PROCESSOR_INFO_MAX_VENDOR_NAME_LENGTH (see page 111)	Maximum length of processor's vendor name.

Structs, Records, Enums

	Name	Description
	NProcessorVendor_ (see page 110)	Enumerates different processor vendors.
	NProcessorVendor (see page 110)	Enumerates different processor vendors.

6.1.2.8.1 Functions

6.1.2.8.1.1 NProcessorInfoGetModelNameA Function

Retrieves processor model name.

C++

```
NResult N_API NProcessorInfoGetModelNameA(NAChar * szValue);
```

Parameters

Parameters	Description
NAChar * szValue	[out] A pointer to memory block that contains processor model name.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	szValue is NULL (see page 133).

Notes

This function is used if compiling for platform without ANSI versions of the functions support. If can not choose between functions NProcessorInfoGetModelNameA and NProcessorInfoGetModelNameW, you can use function without suffix A or W.

Module

NProcessorInfo Module (see page 106)

6.1.2.8.1.2 NProcessorInfoGetModelNameW Function

Retrieves processor model name.

C++

```
NResult N_API NProcessorInfoGetModelNameW(NWChar * szValue);
```

Parameters

Parameters	Description
NWChar * szValue	[out] A pointer to memory block that contains processor model name.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	szValue is NULL (see page 133).

Notes

This function is used if compiling for platforms without Unicode support. If can not choose between functions NProcessorInfoGetModelNameA (see page 106) and NProcessorInfoGetModelNameW, you can use function without suffix A or W.

Module

NProcessorInfo Module (see page 106)

6.1.2.8.1.3 NProcessorInfoGetVendor Function

Retrieves a pointer to enumeration that contains information about processor's vendor.

C++

```
NResult N_API NProcessorInfoGetVendor(NProcessorVendor * pValue);
```

Parameters

Parameters	Description
NProcessorVendor * pValue	[out] A pointer to memory block that contains NProcessorVendor (see page 110) enumeration.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pValue is NULL (see page 133).

Module

NProcessorInfo Module (see page 106)

6.1.2.8.1.4 NProcessorInfoGetVendorNameA Function

Retrieves processor's vendor name.

C++

```
NResult N_API NProcessorInfoGetVendorNameA(NAChar * szValue);
```

Parameters

Parameters	Description
NAChar * szValue	[out] A pointer to memory block that contains processor's vendor name.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	szValue is NULL (see page 133).

Notes

This function is used if compiling for platform without ANSI versions of the functions support. If can not choose between functions NProcessorInfoGetVendorNameA NProcessorInfoGetVendorNameW (see page 108), you can use function without suffix A or W.

Module

NProcessorInfo Module (see page 106)

6.1.2.8.1.5 NProcessorInfoGetVendorNameW Function

Retrieves processor's vendor name.

C++

```
NResult N_API NProcessorInfoGetVendorNameW(NWChar * szValue);
```

Parameters

Parameters	Description
NWChar * szValue	[out] A pointer to memory block that contains processor's vendor name.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	szValue is NULL (see page 133).

Notes

This function is used if compiling for platforms without Unicode support. If can not choose between functions NProcessorInfoGetVendorNameA ([see page 108](#)) NProcessorInfoGetVendorNameW, you can use function without suffix A or W.

Module

NProcessorInfo Module ([see page 106](#))

6.1.2.8.1.6 NProcessorInfoIs3DNowSupported Function**C++**

```
NBool N_API NProcessorInfoIs3DNowSupported();
```

Returns

Returns a boolean value which indicates 3D support for the particular processor model.

Module

NProcessorInfo Module ([see page 106](#))

6.1.2.8.1.7 NProcessorInfoIsMmxSupported Function**C++**

```
NBool N_API NProcessorInfoIsMmxSupported();
```

Returns

Returns a boolean value which indicates if a single instruction set (Mmx) is supported for the particular processor model.

Module

NProcessorInfo Module ([see page 106](#))

6.1.2.8.1.8 NProcessorInfoIsSse2Supported Function**C++**

```
NBool N_API NProcessorInfoIsSse2Supported();
```

Returns

Returns a boolean value which indicates if a Streaming SIMD Extension 2 (SSE2) instruction set is supported for a particular processor model.

Module

NProcessorInfo Module ([see page 106](#))

6.1.2.8.1.9 NProcessorInfoIsSse3Supported Function

C++

```
NBool N_API NProcessorInfoIsSse3Supported();
```

Returns

Returns a boolean value which indicates if a Streaming SIMD Extension 3 (SSE3) instruction set is supported for a particular processor model.

Module

NProcessorInfo Module (see page 106)

6.1.2.8.1.10 NProcessorInfoIsSseSupported Function

C++

```
NBool N_API NProcessorInfoIsSseSupported();
```

Returns

Returns a boolean value which indicates if a Streaming SIMD (SSE) instruction set is supported for a particular processor model.

Module

NProcessorInfo Module (see page 106)

6.1.2.8.2 Structs, Records, Enums

6.1.2.8.2.1 NProcessorVendor Enumeration

Enumerates different processor vendors.

C++

```
typedef enum NProcessorVendor_ {
    npvUnknown = 0,
    npvAmd = 1,
    npvCentaur = 2,
    npvCyrix = 3,
    npvIntel = 4,
    npvNationalSemiconductor = 5,
    npvNexGen = 6,
    npvRiseTechnology = 7,
    npvSiS = 8,
    npvTransmeta = 9,
    npvUmc = 10
} NProcessorVendor;
```

Members

Members	Description
npvUnknown = 0	Vendor is unknown.
npvAmd = 1	Advanced Micro Devices, Inc. (AMD)
npvCentaur = 2	Centaur Technology, Inc.
npvCyrix = 3	Cyrix, Inc.
npvIntel = 4	Intel Corporation.
npvNationalSemiconductor = 5	National Semiconductor Corporation.
npvNexGen = 6	NexGen Technologies, Inc.
npvRiseTechnology = 7	Rise Technology, Inc.

npvSiS = 8	Silicon Integrated Systems Corp.
npvTransmeta = 9	Transmeta Corp.
npvUmc = 10	United Microelectronics Corporation.

Module

NProcessorInfo Module ([see page 106](#))

6.1.2.8.3 Macros**6.1.2.8.3.1 N_PROCESSOR_INFO_MAX_MODEL_NAME_LENGTH Macro**

Maximum length of processor model.

C++

```
#define N_PROCESSOR_INFO_MAX_MODEL_NAME_LENGTH 48
```

Module

NProcessorInfo Module ([see page 106](#))

6.1.2.8.3.2 N_PROCESSOR_INFO_MAX_VENDOR_NAME_LENGTH Macro

Maximum length of processor's vendor name.

C++

```
#define N_PROCESSOR_INFO_MAX_VENDOR_NAME_LENGTH 13
```

Module

NProcessorInfo Module ([see page 106](#))

6.1.2.8.4 Files**6.1.2.8.4.1 NProcessorInfo.h**

Header file for the NProcessorInfo. Provides functionality for getting processor information.

Enumerations

	Name	Description
	NProcessorVendor_ (see page 110)	Enumerates different processor vendors.
	NProcessorVendor (see page 110)	Enumerates different processor vendors.

Functions

	Name	Description
	NProcessorInfoGetModelNameA (see page 106)	Retrieves processor model name.
	NProcessorInfoGetModelNameW (see page 107)	Retrieves processor model name.
	NProcessorInfoGetVendor (see page 107)	Retrieves a pointer to enumeration that contains information about processor's vendor.
	NProcessorInfoGetVendorNameA (see page 108)	Retrieves processor's vendor name.

	NProcessorInfoGetVendorNameW (see page 108)	Retrieves processor's vendor name.
	NProcessorInfoIs3DNowSupported (see page 109)	
	NProcessorInfoIsMmxSupported (see page 109)	
	NProcessorInfoIsSse2Supported (see page 109)	
	NProcessorInfoIsSse3Supported (see page 110)	
	NProcessorInfoIsSseSupported (see page 110)	

Macros

Name	Description
N_PROCESSOR_INFO_MAX_MODEL_NAME_LENGTH (see page 111)	Maximum length of processor model.
N_PROCESSOR_INFO_MAX_VENDOR_NAME_LENGTH (see page 111)	Maximum length of processor's vendor name.

Module

NProcessorInfo Module ([see page 106](#))

6.1.2.9 NTypes Module

Defines types and macros used in Neurotechnology components.

Files

Name	Description
NTypes.h (see page 133)	Header file for NTypes module. Defines types and macros used in Neurotechnology components.

Macros

Name	Description
N_64 (see page 121)	Defined if compiling for 64-bit architecture.
N_DECLARE_HANDLE (see page 121)	Declares handle with specified name.
N_ANSI_C (see page 121)	Defined if ANSI C language compliance is enabled in compiler.
N_BIG_ENDIAN (see page 121)	Defined if compiling for big-endian processor architecture.
N_BYTE_MAX (see page 122)	Maximum value for NByte (see page 116).
N_BYTE_MIN (see page 122)	Minimum value for NByte (see page 116).
N_CALLBACK (see page 122)	Callback.
N_CALLBACK_AW (see page 122)	Picks either ANSI or Unicode (if N_UNICODE (see page 132) is defined) version of the callback (with either 'A' or 'W' suffix accordingly).
N_CPP (see page 122)	Defined if compiling as C++ code.
N_DEBUG (see page 122)	Defined if compiling in debug mode.
N_DOUBLE_EPSILON (see page 123)	Epsilon value for NDouble (see page 117).
N_DOUBLE_MAX (see page 123)	Maximum value for NDouble (see page 117).
N_DOUBLE_MIN (see page 123)	Minimum value for NDouble (see page 117).
N_FAST_FLOAT (see page 123)	Defined if CPU has floating-point instructions.
N_FLOAT_EPSILON (see page 123)	Epsilon value for NFloat (see page 117).
N_FLOAT_MAX (see page 123)	Maximum value for NFloat (see page 117).

N_FLOAT_MIN (see page 124)	Minimum value for NFloat (see page 117).
N_FUNC_AW (see page 124)	Picks either ANSI or Unicode (if N_UNICODE (see page 132) is defined) version of the function (with either 'A' or 'W' suffix accordingly).
N_GCC (see page 124)	Defined if compiling with GCC.
N_INT_MAX (see page 124)	Maximum value for NInt (see page 117).
N_INT_MIN (see page 124)	Minimum value for NInt (see page 117).
N_INT16_MAX (see page 125)	Maximum value for NShort (see page 119).
N_INT16_MIN (see page 125)	Minimum value for NShort (see page 119).
N_INT32_MAX (see page 125)	Maximum value for NInt (see page 117).
N_INT32_MIN (see page 125)	Minimum value for NInt (see page 117).
N_INT64_MAX (see page 125)	Maximum value for NInt64 (see page 118).
N_INT64_MIN (see page 125)	Minimum value for NInt64 (see page 118).
N_INT8_MAX (see page 126)	Maximum value for NSByte (see page 119).
N_INT8_MIN (see page 126)	Minimum value for NSByte (see page 119).
N_LIB (see page 126)	Defined if compiling static library.
N_LINUX (see page 126)	Defined if compiling for Linux.
N_LONG_MAX (see page 126)	Maximum value for NLong (see page 118).
N_LONG_MIN (see page 126)	Minimum value for NLong (see page 118).
N_MAC (see page 127)	Defined if compiling for Mac OS.
N_MSVC (see page 127)	Defined if compiling with Microsoft Visual C++.
N_NO_ANSI_FUNC (see page 127)	Defined if compiling for platform without ANSI versions of the functions support.
N_NO_INT_64 (see page 127)	Defined if compiling for platform without 64-bit integer types support.
N_PACKED (see page 127)	Struct packing/alignment.
N_POS_TYPE_MAX (see page 128)	Maximum value for NPosType (see page 118).
N_POS_TYPE_MIN (see page 128)	Minimum value for NPosType (see page 118).
N_SBYTE_MAX (see page 128)	Maximum value for NSByte (see page 119).
N_SBYTE_MIN (see page 128)	Minimum value for NSByte (see page 119).
N_SHORT_MAX (see page 128)	Maximum value for NShort (see page 119).
N_SHORT_MIN (see page 128)	Minimum value for NShort (see page 119).
N_SINGLE_EPSILON (see page 129)	Epsilon value for NSingle (see page 119).
N_SINGLE_MAX (see page 129)	Maximum value for NSingle (see page 119).
N_SINGLE_MIN (see page 129)	Minimum value for NSingle (see page 119).
N_SIZE_TYPE_MAX (see page 129)	Maximum value for NSizeType (see page 119).
N_SIZE_TYPE_MIN (see page 129)	Minimum value for NSizeType (see page 119).
N_STRUCT_AW (see page 129)	Picks either ANSI or Unicode (if N_UNICODE (see page 132) is defined) version of the struct (with either 'A' or 'W' suffix accordingly)
N_T (see page 130)	Makes either ANSI or Unicode (if N_UNICODE (see page 132) is defined) string or character constant.
N_UINT_MAX (see page 130)	Maximum value for NUInt (see page 119).
N_UINT_MIN (see page 130)	Minimum value for NUInt (see page 119).
N_UINT16_MAX (see page 130)	Maximum value for NUShort (see page 120).
N_UINT16_MIN (see page 130)	Minimum value for NUShort (see page 120).
N_UINT32_MAX (see page 131)	Maximum value for NUInt (see page 119).
N_UINT32_MIN (see page 131)	Minimum value for NUInt (see page 119).
N_UINT64_MAX (see page 131)	Maximum value for NULong (see page 120).
N_UINT64_MIN (see page 131)	Minimum value for NULong (see page 120).
N_UINT8_MAX (see page 131)	Maximum value for NUInt8 (see page 120).

N_UINT8_MIN (see page 131)	Minimum value for NUInt8 (see page 120).
N_ULONG_MAX (see page 132)	Maximum value for NULong (see page 120).
N_ULONG_MIN (see page 132)	Minimum value for NULong (see page 120).
N_UNICODE (see page 132)	Defined if compiling with Unicode character set (affects NChar (see page 117) type).
N_USHORT_MAX (see page 132)	Maximum value for NUShort (see page 120).
N_USHORT_MIN (see page 132)	Minimum value for NUShort (see page 120).
N_WCHAR_SIZE (see page 132)	!defined(N_WINDOWS (see page 133)) && (!defined(__SIZEOF_WCHAR_T__) __SIZEOF_WCHAR_T__ != 2)
N_WINDOWS (see page 133)	Defined if compiling for Windows.
NFalse (see page 133)	False value for NBoolean (see page 116).
NTrue (see page 133)	True value for NBoolean (see page 116).
NULL (see page 133)	Null value for pointer.

Structs, Records, Enums

	Name	Description
	NIndexPair_ (see page 115)	Represents a pair of indexes.
	NRational_ (see page 115)	Represents a signed rational number.
	NURational_ (see page 115)	Represents an unsigned rational number.
	NIndexPair (see page 115)	Represents a pair of indexes.
	NRational (see page 115)	Represents a signed rational number.
	NURational (see page 115)	Represents an unsigned rational number.

Types

Name	Description
NChar (see page 116)	ANSI character (8-bit).
NByte (see page 116)	Same as NUInt8 (see page 120).
NBool (see page 116)	Same as NBoolean (see page 116).
NBoolean (see page 116)	32-bit boolean value. See also NTrue (see page 133) and NFalse (see page 133).
NChar (see page 117)	!N_NO_UNICODE Character type. Either NChar (see page 116) or NWChar (see page 121) (if N_UNICODE (see page 132) is defined).
NDouble (see page 117)	Double precision floating point number.
NFloat (see page 117)	Same as NSingle (see page 119).
NHandle (see page 117)	Pointer to unspecified data (same as void *).
NInt (see page 117)	Same as NInt32 (see page 118).
NInt16 (see page 117)	16-bit signed integer (short).
NInt32 (see page 118)	32-bit signed integer (int).
NInt64 (see page 118)	64-bit signed integer (long). Not available on some 32-bit platforms.
NInt8 (see page 118)	8-bit signed integer (signed byte).
NLong (see page 118)	Same as NInt64 (see page 118).
NPosType (see page 118)	Platform dependent position type. Signed 64-bit (or 32-bit on some platforms) integer on 32-bit platform, signed 64-bit integer on 64-bit platform).
NResult (see page 118)	Result of a function (same as NInt (see page 117)). See also NErrors module.
NSByte (see page 119)	Same as NInt8 (see page 118).
NShort (see page 119)	Same as NInt16 (see page 117).
NSingle (see page 119)	Single precision floating point number.

NSizeType (see page 119)	Platform dependent size type. Unsigned 32-bit integer on 32-bit platform, unsigned 64-bit integer on 64-bit platform.
NUInt (see page 119)	Same as NUInt32 (see page 120).
NUInt16 (see page 120)	16-bit unsigned integer (unsigned short).
NUInt32 (see page 120)	32-bit unsigned integer (unsigned int).
NUInt64 (see page 120)	64-bit unsigned integer (unsigned long). Not available on some 32-bit platforms.
NUInt8 (see page 120)	8-bit unsigned integer (byte).
NULong (see page 120)	Same as NUInt64 (see page 120).
NUShort (see page 120)	Same as NUInt16 (see page 120).
NWChar (see page 121)	Unicode character (16-bit).

6.1.2.9.1 Structs, Records, Enums

6.1.2.9.1.1 NIndexPair Structure

Represents a pair of indexes.

C++

```
typedef struct NIndexPair_ {
    NInt Index1;
    NInt Index2;
} NIndexPair;
```

Members

Members	Description
NInt Index1;	First index of this NIndexPair.
NInt Index2;	Second index of this NIndexPair.

Module

NTypes Module ([see page 112](#))

6.1.2.9.1.2 NRational Structure

Represents a signed rational number.

C++

```
typedef struct NRational_ {
    NInt Numerator;
    NInt Denominator;
} NRational;
```

Members

Members	Description
NInt Numerator;	Numerator of this NRational.
NInt Denominator;	Denominator of this NRational.

Module

NTypes Module ([see page 112](#))

6.1.2.9.1.3 NURational Structure

Represents an unsigned rational number.

C++

```
typedef struct NURational_ {  
    NUInt Numerator;  
    NUInt Denominator;  
} NURational;
```

Members

Members	Description
NUInt Numerator;	Numerator of this NURational.
NUInt Denominator;	Denominator of this NURational.

Module

NTypes Module ([see page 112](#))

6.1.2.9.2 Types

6.1.2.9.2.1 NChar Type

ANSI character (8-bit).

C++

```
typedef char NChar;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.2 NByte Type

Same as NUInt8 ([see page 120](#)).

C++

```
typedef NUInt8 NByte;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.3 NBool Type

Same as NBoolean ([see page 116](#)).

C++

```
typedef NBoolean NBool;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.4 NBoolean Type

32-bit boolean value. See also NTrue ([see page 133](#)) and NFalse ([see page 133](#)).

C++

```
typedef int NBoolean;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.5 NChar Type

!N_NO_UNICODE

Character type. Either NChar (see page 116) or NWChar (see page 121) (if N_UNICODE (see page 132) is defined).

C++

```
typedef NChar NChar;
```

Module

NTypes Module (see page 112)

6.1.2.9.2.6 NDouble Type

Double precision floating point number.

C++

```
typedef double NDouble;
```

Module

NTypes Module (see page 112)

6.1.2.9.2.7 NFloat Type

Same as NSingle (see page 119).

C++

```
typedef NSingle NFloat;
```

Module

NTypes Module (see page 112)

6.1.2.9.2.8 NHandle Type

Pointer to unspecified data (same as void *).

C++

```
typedef void * NHandle;
```

Module

NTypes Module (see page 112)

6.1.2.9.2.9 NInt Type

Same as NInt32 (see page 118).

C++

```
typedef NInt32 NInt;
```

Module

NTypes Module (see page 112)

6.1.2.9.2.10 NInt16 Type

16-bit signed integer (short).

C++

```
typedef signed short NInt16;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.11 NInt32 Type

32-bit signed integer (int).

C++

```
typedef signed int NInt32;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.12 NInt64 Type

64-bit signed integer (long). Not available on some 32-bit platforms.

C++

```
typedef signed long long NInt64;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.13 NInt8 Type

8-bit signed integer (signed byte).

C++

```
typedef signed char NInt8;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.14 NLong Type

Same as NInt64 ([see page 118](#)).

C++

```
typedef NInt64 NLong;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.15 NPosType Type

Platform dependent position type. Signed 64-bit (or 32-bit on some platforms) integer on 32-bit platform, signed 64-bit integer on 64-bit platform).

C++

```
typedef NInt64 NPosType;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.16 NResult Type

Result of a function (same as NInt ([see page 117](#))). See also NErrors module.

C++

```
typedef NInt NResult;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.17 NSByte Type

Same as NInt8 ([see page 118](#)).

C++

```
typedef NInt8 NSByte;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.18 NShort Type

Same as NInt16 ([see page 117](#)).

C++

```
typedef NInt16 NShort;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.19 NSingle Type

Single precision floating point number.

C++

```
typedef float NSingle;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.20 NSizeType Type

Platform dependent size type. Unsigned 32-bit integer on 32-bit platform, unsigned 64-bit integer on 64-bit platform.

C++

```
typedef NUInt32 NSizeType;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.21 NUInt Type

Same as NUInt32 ([see page 120](#)).

C++

```
typedef NUInt32 NUInt;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.22 NUInt16 Type

16-bit unsigned integer (unsigned short).

C++

```
typedef unsigned short NUInt16;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.23 NUInt32 Type

32-bit unsigned integer (unsigned int).

C++

```
typedef unsigned int NUInt32;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.24 NUInt64 Type

64-bit unsigned integer (unsigned long). Not available on some 32-bit platforms.

C++

```
typedef unsigned long long NUInt64;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.25 NUInt8 Type

8-bit unsigned integer (byte).

C++

```
typedef unsigned char NUInt8;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.26 NULong Type

Same as NUInt64 ([see page 120](#)).

C++

```
typedef NUInt64 NULong;
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.2.27 NUShort Type

Same as NUInt16 ([see page 120](#)).

C++

```
typedef NUInt16 NUShort;
```

Module

NTypes Module (see page 112)

6.1.2.9.2.28 NWChar Type

Unicode character (16-bit).

C++

```
typedef int NWChar;
```

Module

NTypes Module (see page 112)

6.1.2.9.3 Macros**6.1.2.9.3.1 N_64 Macro**

Defined if compiling for 64-bit architecture.

C++

```
#define N_64
```

Module

NTypes Module (see page 112)

6.1.2.9.3.2 N_DECLARE_HANDLE Macro

Declares handle with specified name.

C++

```
#define N_DECLARE_HANDLE(name) typedef struct name##_ { int unused; } * name;
```

Module

NTypes Module (see page 112)

6.1.2.9.3.3 N_ANSI_C Macro

Defined if ANSI C language compliance is enabled in compiler.

C++

```
#define N_ANSI_C
```

Module

NTypes Module (see page 112)

6.1.2.9.3.4 N_BIG_ENDIAN Macro

Defined if compiling for big-endian processor architecture.

C++

```
#define N_BIG_ENDIAN
```

Module

NTypes Module (see page 112)

6.1.2.9.3.5 N_BYTE_MAX Macro

Maximum value for NByte ([↗](#) see page 116).

C++

```
#define N_BYTE_MAX N_UINT8_MAX
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.6 N_BYTE_MIN Macro

Minimum value for NByte ([↗](#) see page 116).

C++

```
#define N_BYTE_MIN N_UINT8_MIN
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.7 N_CALLBACK Macro

Callback.

C++

```
#define N_CALLBACK N_CALL_CONV *
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.8 N_CALLBACK_AW Macro

Picks either ANSI or Unicode (if N_UNICODE ([↗](#) see page 132) is defined) version of the callback (with either 'A' or 'W' suffix accordingly).

C++

```
#define N_CALLBACK_AW(name) name##A
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.9 N_CPP Macro

Defined if compiling as C++ code.

C++

```
#define N_CPP
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.10 N_DEBUG Macro

Defined if compiling in debug mode.

C++

```
#define N_DEBUG
```

Module

NTypes Module (see page 112)

6.1.2.9.3.11 N_DOUBLE_EPSILON Macro

Epsilon value for NDouble (see page 117).

C++

```
#define N_DOUBLE_EPSILON 2.2204460492503131e-016
```

Module

NTypes Module (see page 112)

6.1.2.9.3.12 N_DOUBLE_MAX Macro

Maximum value for NDouble (see page 117).

C++

```
#define N_DOUBLE_MAX 1.7976931348623158e+308
```

Module

NTypes Module (see page 112)

6.1.2.9.3.13 N_DOUBLE_MIN Macro

Minimum value for NDouble (see page 117).

C++

```
#define N_DOUBLE_MIN 2.2250738585072014e-308
```

Module

NTypes Module (see page 112)

6.1.2.9.3.14 N_FAST_FLOAT Macro

Defined if CPU has floating-point instructions.

C++

```
#define N_FAST_FLOAT
```

Module

NTypes Module (see page 112)

6.1.2.9.3.15 N_FLOAT_EPSILON Macro

Epsilon value for NFloat (see page 117).

C++

```
#define N_FLOAT_EPSILON N_SINGLE_EPSILON
```

Module

NTypes Module (see page 112)

6.1.2.9.3.16 N_FLOAT_MAX Macro

Maximum value for NFloat (see page 117).

C++

```
#define N_FLOAT_MAX N_SINGLE_MAX
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.17 N_FLOAT_MIN Macro

Minimum value for NFloat ([see page 117](#)).

C++

```
#define N_FLOAT_MIN N_SINGLE_MIN
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.18 N_FUNC_AW Macro

Picks either ANSI or Unicode (if N_UNICODE ([see page 132](#)) is defined) version of the function (with either 'A' or 'W' suffix accordingly).

C++

```
#define N_FUNC_AW(name) name##A
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.19 N_GCC Macro

Defined if compiling with GCC.

C++

```
#define N_GCC
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.20 N_INT_MAX Macro

Maximum value for NInt ([see page 117](#)).

C++

```
#define N_INT_MAX N_INT32_MAX
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.21 N_INT_MIN Macro

Minimum value for NInt ([see page 117](#)).

C++

```
#define N_INT_MIN N_INT32_MIN
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.22 N_INT16_MAX Macro

Maximum value for NShort ([↗](#) see page 119).

C++

```
#define N_INT16_MAX ((NInt16)0x7FFF)
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.23 N_INT16_MIN Macro

Minimum value for NShort ([↗](#) see page 119).

C++

```
#define N_INT16_MIN ((NInt16)0x8000)
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.24 N_INT32_MAX Macro

Maximum value for NInt ([↗](#) see page 117).

C++

```
#define N_INT32_MAX 0x7FFFFFFF
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.25 N_INT32_MIN Macro

Minimum value for NInt ([↗](#) see page 117).

C++

```
#define N_INT32_MIN 0x80000000
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.26 N_INT64_MAX Macro

Maximum value for NInt64 ([↗](#) see page 118).

C++

```
#define N_INT64_MAX 0x7FFFFFFFFFFFFFFFll
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.27 N_INT64_MIN Macro

Minimum value for NInt64 ([↗](#) see page 118).

C++

```
#define N_INT64_MIN 0x8000000000000000ll
```

Module

NTypes Module (see page 112)

6.1.2.9.3.28 N_INT8_MAX Macro

Maximum value for NSByte (see page 119).

C++

```
#define N_INT8_MAX ((NInt8)0x7F)
```

Module

NTypes Module (see page 112)

6.1.2.9.3.29 N_INT8_MIN Macro

Minimum value for NSByte (see page 119).

C++

```
#define N_INT8_MIN ((NInt8)0x80)
```

Module

NTypes Module (see page 112)

6.1.2.9.3.30 N_LIB Macro

Defined if compiling static library.

C++

```
#define N_LIB
```

Module

NTypes Module (see page 112)

6.1.2.9.3.31 N_LINUX Macro

Defined if compiling for Linux.

C++

```
#define N_LINUX
```

Module

NTypes Module (see page 112)

6.1.2.9.3.32 N_LONG_MAX Macro

Maximum value for NLong (see page 118).

C++

```
#define N_LONG_MAX N_INT64_MAX
```

Module

NTypes Module (see page 112)

6.1.2.9.3.33 N_LONG_MIN Macro

Minimum value for NLong (see page 118).

C++

```
#define N_LONG_MIN N_INT64_MIN
```

Module

NTypes Module (see page 112)

6.1.2.9.3.34 N_MAC Macro

Defined if compiling for Mac OS.

C++

```
#define N_MAC
```

Module

NTypes Module (see page 112)

6.1.2.9.3.35 N_MSVC Macro

Defined if compiling with Microsoft Visual C++.

C++

```
#define N_MSVC
```

Module

NTypes Module (see page 112)

6.1.2.9.3.36 N_NO_ANSI_FUNC Macro

Defined if compiling for platform without ANSI versions of the functions support.

C++

```
#define N_NO_ANSI_FUNC
```

Module

NTypes Module (see page 112)

6.1.2.9.3.37 N_NO_INT_64 Macro

Defined if compiling for platform without 64-bit integer types support.

C++

```
#define N_NO_INT_64
```

Module

NTypes Module (see page 112)

6.1.2.9.3.38 N_PACKED Macro

Struct packing/alignment.

C++

```
#define N_PACKED
```

Module

NTypes Module (see page 112)

6.1.2.9.3.39 N_POS_TYPE_MAX Macro

Maximum value for NPosType ([↗](#) see page 118).

C++

```
#define N_POS_TYPE_MAX N_INT64_MAX
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.40 N_POS_TYPE_MIN Macro

Minimum value for NPosType ([↗](#) see page 118).

C++

```
#define N_POS_TYPE_MIN N_INT64_MIN
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.41 N_SBYTE_MAX Macro

Maximum value for NSByte ([↗](#) see page 119).

C++

```
#define N_SBYTE_MAX N_INT8_MAX
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.42 N_SBYTE_MIN Macro

Minimum value for NSByte ([↗](#) see page 119).

C++

```
#define N_SBYTE_MIN N_INT8_MIN
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.43 N_SHORT_MAX Macro

Maximum value for NShort ([↗](#) see page 119).

C++

```
#define N_SHORT_MAX N_INT16_MAX
```

Module

NTypes Module ([↗](#) see page 112)

6.1.2.9.3.44 N_SHORT_MIN Macro

Minimum value for NShort ([↗](#) see page 119).

C++

```
#define N_SHORT_MIN N_INT16_MIN
```

Module

NTypes Module (see page 112)

6.1.2.9.3.45 N_SINGLE_EPSILON Macro

Epsilon value for NSingle (see page 119).

C++

```
#define N_SINGLE_EPSILON 1.192092896e-07F
```

Module

NTypes Module (see page 112)

6.1.2.9.3.46 N_SINGLE_MAX Macro

Maximum value for NSingle (see page 119).

C++

```
#define N_SINGLE_MAX 3.402823466e+38F
```

Module

NTypes Module (see page 112)

6.1.2.9.3.47 N_SINGLE_MIN Macro

Minimum value for NSingle (see page 119).

C++

```
#define N_SINGLE_MIN 1.175494351e-38F
```

Module

NTypes Module (see page 112)

6.1.2.9.3.48 N_SIZE_TYPE_MAX Macro

Maximum value for NSizeType (see page 119).

C++

```
#define N_SIZE_TYPE_MAX N_UINT32_MAX
```

Module

NTypes Module (see page 112)

6.1.2.9.3.49 N_SIZE_TYPE_MIN Macro

Minimum value for NSizeType (see page 119).

C++

```
#define N_SIZE_TYPE_MIN N_UINT32_MIN
```

Module

NTypes Module (see page 112)

6.1.2.9.3.50 N_STRUCT_AW Macro

Picks either ANSI or Unicode (if N_UNICODE (see page 132) is defined) version of the struct (with either 'A' or 'W' suffix accordingly)

C++

```
#define N_STRUCT_AW(name) name##A
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.51 N_T Macro

Makes either ANSI or Unicode (if N_UNICODE ([see page 132](#)) is defined) string or character constant.

C++

```
#define N_T(text) N_T_(text)
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.52 N_UINT_MAX Macro

Maximum value for NUInt ([see page 119](#)).

C++

```
#define N_UINT_MAX N_UINT32_MAX
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.53 N_UINT_MIN Macro

Minimum value for NUInt ([see page 119](#)).

C++

```
#define N_UINT_MIN N_UINT32_MIN
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.54 N_UINT16_MAX Macro

Maximum value for NUShort ([see page 120](#)).

C++

```
#define N_UINT16_MAX ((NUInt16)0xFFFFu)
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.55 N_UINT16_MIN Macro

Minimum value for NUShort ([see page 120](#)).

C++

```
#define N_UINT16_MIN ((NUInt16)0x0000u)
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.56 N_UINT32_MAX Macro

Maximum value for NUInt (see page 119).

C++

```
#define N_UINT32_MAX 0xFFFFFFFFu
```

Module

NTypes Module (see page 112)

6.1.2.9.3.57 N_UINT32_MIN Macro

Minimum value for NUInt (see page 119).

C++

```
#define N_UINT32_MIN 0x00000000u
```

Module

NTypes Module (see page 112)

6.1.2.9.3.58 N_UINT64_MAX Macro

Maximum value for NULong (see page 120).

C++

```
#define N_UINT64_MAX 0xFFFFFFFFFFFFFFFFull
```

Module

NTypes Module (see page 112)

6.1.2.9.3.59 N_UINT64_MIN Macro

Minimum value for NULong (see page 120).

C++

```
#define N_UINT64_MIN 0x0000000000000000ull
```

Module

NTypes Module (see page 112)

6.1.2.9.3.60 N_UINT8_MAX Macro

Maximum value for NUInt8 (see page 120).

C++

```
#define N_UINT8_MAX ((NUInt8)0xFFu)
```

Module

NTypes Module (see page 112)

6.1.2.9.3.61 N_UINT8_MIN Macro

Minimum value for NUInt8 (see page 120).

C++

```
#define N_UINT8_MIN ((NUInt8)0x00u)
```

Module

NTypes Module (see page 112)

6.1.2.9.3.62 N_ULONG_MAX Macro

Maximum value for NULong (see page 120).

C++

```
#define N_ULONG_MAX N_UINT64_MAX
```

Module

NTypes Module (see page 112)

6.1.2.9.3.63 N_ULONG_MIN Macro

Minimum value for NULong (see page 120).

C++

```
#define N_ULONG_MIN N_UINT64_MIN
```

Module

NTypes Module (see page 112)

6.1.2.9.3.64 N_UNICODE Macro

Defined if compiling with Unicode character set (affects NChar (see page 117) type).

C++

```
#define N_UNICODE
```

Module

NTypes Module (see page 112)

6.1.2.9.3.65 N_USHORT_MAX Macro

Maximum value for NUShort (see page 120).

C++

```
#define N_USHORT_MAX N_UINT16_MAX
```

Module

NTypes Module (see page 112)

6.1.2.9.3.66 N_USHORT_MIN Macro

Minimum value for NUShort (see page 120).

C++

```
#define N_USHORT_MIN N_UINT16_MIN
```

Module

NTypes Module (see page 112)

6.1.2.9.3.67 N_WCHAR_SIZE Macro

```
!defined(N_WINDOWS) && (!defined(__SIZEOF_WCHAR_T__) || __SIZEOF_WCHAR_T__ != 2)
```

C++

```
#define N_WCHAR_SIZE 4
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.68 N_WINDOWS Macro

Defined if compiling for Windows.

C++

```
#define N_WINDOWS
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.69 NFalse Macro

False value for NBoolean ([see page 116](#)).

C++

```
#define NFalse 0
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.70 NTrue Macro

True value for NBoolean ([see page 116](#)).

C++

```
#define NTrue 1
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.3.71 NULL Macro

Null value for pointer.

C++

```
#define NULL 0
```

Module

NTypes Module ([see page 112](#))

6.1.2.9.4 Files**6.1.2.9.4.1 NTypes.h**

Header file for NTypes module. Defines types and macros used in Neurotechnology components.

Macros

Name	Description
N_64 (see page 121)	Defined if compiling for 64-bit architecture.

N_ANSI_C (see page 121)	Defined if ANSI C language compliance is enabled in compiler.
N_BIG_ENDIAN (see page 121)	Defined if compiling for big-endian processor architecture.
N_BYTE_MAX (see page 122)	Maximum value for NByte (see page 116).
N_BYTE_MIN (see page 122)	Minimum value for NByte (see page 116).
N_CALLBACK (see page 122)	Callback.
N_CALLBACK_AW (see page 122)	Picks either ANSI or Unicode (if N_UNICODE (see page 132) is defined) version of the callback (with either 'A' or 'W' suffix accordingly).
N_CPP (see page 122)	Defined if compiling as C++ code.
N_DEBUG (see page 122)	Defined if compiling in debug mode.
N_DECLARE_HANDLE (see page 121)	Declares handle with specified name.
N_DOUBLE_EPSILON (see page 123)	Epsilon value for NDouble (see page 117).
N_DOUBLE_MAX (see page 123)	Maximum value for NDouble (see page 117).
N_DOUBLE_MIN (see page 123)	Minimum value for NDouble (see page 117).
N_FAST_FLOAT (see page 123)	Defined if CPU has floating-point instructions.
N_FLOAT_EPSILON (see page 123)	Epsilon value for NFloat (see page 117).
N_FLOAT_MAX (see page 123)	Maximum value for NFloat (see page 117).
N_FLOAT_MIN (see page 124)	Minimum value for NFloat (see page 117).
N_FUNC_AW (see page 124)	Picks either ANSI or Unicode (if N_UNICODE (see page 132) is defined) version of the function (with either 'A' or 'W' suffix accordingly).
N_GCC (see page 124)	Defined if compiling with GCC.
N_INT_MAX (see page 124)	Maximum value for NInt (see page 117).
N_INT_MIN (see page 124)	Minimum value for NInt (see page 117).
N_INT16_MAX (see page 125)	Maximum value for NShort (see page 119).
N_INT16_MIN (see page 125)	Minimum value for NShort (see page 119).
N_INT32_MAX (see page 125)	Maximum value for NInt (see page 117).
N_INT32_MIN (see page 125)	Minimum value for NInt (see page 117).
N_INT64_MAX (see page 125)	Maximum value for NInt64 (see page 118).
N_INT64_MIN (see page 125)	Minimum value for NInt64 (see page 118).
N_INT8_MAX (see page 126)	Maximum value for NSByte (see page 119).
N_INT8_MIN (see page 126)	Minimum value for NSByte (see page 119).
N_LIB (see page 126)	Defined if compiling static library.
N_LINUX (see page 126)	Defined if compiling for Linux.
N_LONG_MAX (see page 126)	Maximum value for NLong (see page 118).
N_LONG_MIN (see page 126)	Minimum value for NLong (see page 118).
N_MAC (see page 127)	Defined if compiling for Mac OS.
N_MSVC (see page 127)	Defined if compiling with Microsoft Visual C++.
N_NO_ANSI_FUNC (see page 127)	Defined if compiling for platform without ANSI versions of the functions support.
N_NO_INT_64 (see page 127)	Defined if compiling for platform without 64-bit integer types support.
N_PACKED (see page 127)	Struct packing/alignment.
N_POS_TYPE_MAX (see page 128)	Maximum value for NPosType (see page 118).
N_POS_TYPE_MIN (see page 128)	Minimum value for NPosType (see page 118).
N_SBYTE_MAX (see page 128)	Maximum value for NSByte (see page 119).
N_SBYTE_MIN (see page 128)	Minimum value for NSByte (see page 119).
N_SHORT_MAX (see page 128)	Maximum value for NShort (see page 119).
N_SHORT_MIN (see page 128)	Minimum value for NShort (see page 119).
N_SINGLE_EPSILON (see page 129)	Epsilon value for NSingle (see page 119).
N_SINGLE_MAX (see page 129)	Maximum value for NSingle (see page 119).

N_SINGLE_MIN (see page 129)	Minimum value for NSingle (see page 119).
N_SIZE_TYPE_MAX (see page 129)	Maximum value for NSizeType (see page 119).
N_SIZE_TYPE_MIN (see page 129)	Minimum value for NSizeType (see page 119).
N_STRUCT_AW (see page 129)	Picks either ANSI or Unicode (if N_UNICODE (see page 132) is defined) version of the struct (with either 'A' or 'W' suffix accordingly)
N_T (see page 130)	Makes either ANSI or Unicode (if N_UNICODE (see page 132) is defined) string or character constant.
N_UINT_MAX (see page 130)	Maximum value for NUInt (see page 119).
N_UINT_MIN (see page 130)	Minimum value for NUInt (see page 119).
N_UINT16_MAX (see page 130)	Maximum value for NUShort (see page 120).
N_UINT16_MIN (see page 130)	Minimum value for NUShort (see page 120).
N_UINT32_MAX (see page 131)	Maximum value for NUInt (see page 119).
N_UINT32_MIN (see page 131)	Minimum value for NUInt (see page 119).
N_UINT64_MAX (see page 131)	Maximum value for NULong (see page 120).
N_UINT64_MIN (see page 131)	Minimum value for NULong (see page 120).
N_UINT8_MAX (see page 131)	Maximum value for NUInt8 (see page 120).
N_UINT8_MIN (see page 131)	Minimum value for NUInt8 (see page 120).
N_ULONG_MAX (see page 132)	Maximum value for NULong (see page 120).
N_ULONG_MIN (see page 132)	Minimum value for NULong (see page 120).
N_UNICODE (see page 132)	Defined if compiling with Unicode character set (affects NChar (see page 117) type).
N_USHORT_MAX (see page 132)	Maximum value for NUShort (see page 120).
N_USHORT_MIN (see page 132)	Minimum value for NUShort (see page 120).
N_WCHAR_SIZE (see page 132)	!defined(N_WINDOWS (see page 133)) && (!defined(__SIZEOF_WCHAR_T__) __SIZEOF_WCHAR_T__ != 2)
N_WINDOWS (see page 133)	Defined if compiling for Windows.
NFalse (see page 133)	False value for NBoolean (see page 116).
NTrue (see page 133)	True value for NBoolean (see page 116).
NULL (see page 133)	Null value for pointer.

Module

NTypes Module (see page 112)

Structures

	Name	Description
	NIndexPair_ (see page 115)	Represents a pair of indexes.
	NRational_ (see page 115)	Represents a signed rational number.
	NURational_ (see page 115)	Represents an unsigned rational number.
	NIndexPair (see page 115)	Represents a pair of indexes.
	NRational (see page 115)	Represents a signed rational number.
	NURational (see page 115)	Represents an unsigned rational number.

Types

Name	Description
NChar (see page 116)	ANSI character (8-bit).
NByte (see page 116)	Same as NUInt8 (see page 120).
NBool (see page 116)	Same as NBoolean (see page 116).
NBoolean (see page 116)	32-bit boolean value. See also NTrue (see page 133) and NFalse (see page 133).

NChar (see page 117)	!N_NO_UNICODE Character type. Either NChar (see page 116) or NWChar (see page 121) (if N_UNICODE (see page 132) is defined).
NDouble (see page 117)	Double precision floating point number.
NFloat (see page 117)	Same as NSingle (see page 119).
NHandle (see page 117)	Pointer to unspecified data (same as void *).
NInt (see page 117)	Same as NInt32 (see page 118).
NInt16 (see page 117)	16-bit signed integer (short).
NInt32 (see page 118)	32-bit signed integer (int).
NInt64 (see page 118)	64-bit signed integer (long). Not available on some 32-bit platforms.
NInt8 (see page 118)	8-bit signed integer (signed byte).
NLong (see page 118)	Same as NInt64 (see page 118).
NPosType (see page 118)	Platform dependent position type. Signed 64-bit (or 32-bit on some platforms) integer on 32-bit platform, signed 64-bit integer on 64-bit platform).
NResult (see page 118)	Result of a function (same as NInt (see page 117)). See also NErrors module.
NSByte (see page 119)	Same as NInt8 (see page 118).
NShort (see page 119)	Same as NInt16 (see page 117).
NSingle (see page 119)	Single precision floating point number.
NSizeType (see page 119)	Platform dependent size type. Unsigned 32-bit integer on 32-bit platform, unsigned 64-bit integer on 64-bit platform.
NUInt (see page 119)	Same as NUInt32 (see page 120).
NUInt16 (see page 120)	16-bit unsigned integer (unsigned short).
NUInt32 (see page 120)	32-bit unsigned integer (unsigned int).
NUInt64 (see page 120)	64-bit unsigned integer (unsigned long). Not available on some 32-bit platforms.
NUInt8 (see page 120)	8-bit unsigned integer (byte).
NULong (see page 120)	Same as NUInt64 (see page 120).
NUShort (see page 120)	Same as NUInt16 (see page 120).
NWChar (see page 121)	Unicode character (16-bit).

6.1.3 NDeviceManager Library

Provides functionality for managing devices, like cameras or fingerprint scanners.

Remarks

Requirements (Windows)

- **Import Library:** NDeviceManager.dll.lib
- **DLL:** NDeviceManager.dll
- **Required DLLs:**
 - NCore.dll (see page 74).
 - NImages.dll (see page 188)

Requirements (Linux)

- **Shared object:** libNDeviceManager.so.
- **Required objects:**

- libNCore.so.
- libNImages.so.

Modules

Name	Description
FPScanner Module (↗ see page 137)	One instance represents one physical device.
FPScannerMan Module (↗ see page 143)	Scanners manager enumerates, creates scanners.
NDeviceManager Module (↗ see page 151)	Provides functions for retrieving information about NDeviceManager library.

6.1.3.1 FPScanner Module

One instance represents one physical device.

Files

Name	Description
FPScanner.h (↗ see page 142)	Header file for FPScanner module.

Functions

	Name	Description
⇒	FPScannerGetIdA (↗ see page 137)	Gets associated device identifier.
⇒	FPScannerGetIdW (↗ see page 138)	Gets associated device identifier.
⇒	FPScannerIsCapturing (↗ see page 138)	Checks scanner status.
⇒	FPScannerSetFingerPlacedCallback (↗ see page 139)	Sets a callback function that is called immediately after finger is placed on a scanner.
⇒	FPScannerSetFingerRemovedCallback (↗ see page 139)	Sets a callback function that is called immediately after finger is removed from a scanner.
⇒	FPScannerSetImageScannedCallback (↗ see page 140)	Sets a callback function that is called immediately after image is scanned.
⇒	FPScannerSetIsCapturingChangedCallback (↗ see page 140)	Sets a callback function that is called immediately capturing state is changed.
⇒	FPScannerStartCapturing (↗ see page 140)	Starts capturing fingerprint image from certain device.
⇒	FPScannerStartCapturingForOneImage (↗ see page 141)	Starts capturing one fingerprint image from certain device.
⇒	FPScannerStopCapturing (↗ see page 141)	Stops capturing fingerprint image.

Variables

Name	Description
hImage (↗ see page 142)	Handle to the image that has been scanned.

6.1.3.1.1 Functions

6.1.3.1.1.1 FPScannerGetIdA Function

Gets associated device identifier.

C++

```
NResult N_API FPScannerGetIdA(HFPScanner hScanner, NACHar * pValue);
```

Parameters

Parameters	Description
HFPScanner hScanner	[in] Handle to the FPScanner (see page 137) object.
NChar * pValue	[out] Pointer to string that receives scanner identifier.

Returns

If the function succeeds and pValue is NULL (see page 133), the return value is length of the string (not including the NULL (see page 133)-terminator) pValue should point to.

If the function succeeds and pValue is not NULL (see page 133), the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	hScanner (see page 150) is NULL (see page 133).

Remarks

This function is used if compiling for platform without ANSI versions of the functions support.

Module

FPScanner Module (see page 137)

6.1.3.1.1.2 FPScannerGetIdW Function

Gets associated device identifier.

C++

```
NResult N_API FPScannerGetIdW(HFPScanner hScanner, NChar * pValue);
```

Parameters

Parameters	Description
HFPScanner hScanner	[in] Handle to the FPScanner (see page 137) object.
NChar * pValue	[out] Pointer to string that receives scanner identifier.

Returns

If the function succeeds and pValue is NULL (see page 133), the return value is length of the string (not including the NULL (see page 133)-terminator) pValue should point to.

If the function succeeds and pValue is not NULL (see page 133), the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	hScanner (see page 150) is NULL (see page 133).

Remarks

This function is used if compiling for platforms without Unicode support.

Module

FPScanner Module (see page 137)

6.1.3.1.1.3 FPScannerIsCapturing Function

Checks scanner status.

C++

```
NResult N_API FPScannerIsCapturing(HFPScanner hScanner, NBool * pValue);
```

Parameters

Parameters	Description
HFPScanner hScanner	[in] Handle to the FPScanner (see page 137) object.
NBool * pValue	[out] Pointer to NBool (see page 116) that receives value indicating whether scanner is already capturing.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hScanner</i> (see page 150) or <i>pValue</i> is NULL (see page 133).

Module

FPScanner Module (see page 137)

6.1.3.1.1.4 FPScannerSetFingerPlacedCallback Function

Sets a callback function that is called immediately after finger is placed on a scanner.

C++

```
NResult N_API FPScannerSetFingerPlacedCallback(HFPScanner hScanner, FPScannerCallback pCallback, void * pParam);
```

Parameters

Parameters	Description
HFPScanner hScanner	Handle to the FPScanner (see page 137)
FPScannerCallback pCallback	Address of the function that is called after finger is placed on a scanner.
void * pParam	Parameter that is passed to the callback function.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

FPScanner Module (see page 137)

6.1.3.1.1.5 FPScannerSetFingerRemovedCallback Function

Sets a callback function that is called immediately after finger is removed from a scanner.

C++

```
NResult N_API FPScannerSetFingerRemovedCallback(HFPScanner hScanner, FPScannerCallback pCallback, void * pParam);
```

Parameters

Parameters	Description
HFPScanner hScanner	Handle to the FPScanner (see page 137)
FPScannerCallback pCallback	Address of the function that is called after finger is removed from a scanner.

void * pParam	Parameter that is passed to the callback function.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

FPScanner Module (see page 137)

6.1.3.1.1.6 FPScannerSetImageScannedCallback Function

Sets a callback function that is called immediately after image is scanned.

C++

```
NResult N_API FPScannerSetImageScannedCallback(HFPScanner hScanner,
FPScannerImageScannedCallback pCallback, void * pParam);
```

Parameters

Parameters	Description
HFPScanner hScanner	Handle to the FPScanner (see page 137)
FPScannerImageScannedCallback pCallback	Address of the function that is called after image is scanned.
void * pParam	Parameter that is passed to the callback function.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

FPScanner Module (see page 137)

6.1.3.1.1.7 FPScannerSetIsCapturingChangedCallback Function

Sets a callback function that is called immediately capturing state is changed.

C++

```
NResult N_API FPScannerSetIsCapturingChangedCallback(HFPScanner hScanner, FPScannerCallback
pCallback, void * pParam);
```

Parameters

Parameters	Description
HFPScanner hScanner	Handle to the FPScanner (see page 137)
FPScannerCallback pCallback	Address of the function that is called after capturing state is changed.
void * pParam	Parameter that is passed to the callback function.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

FPScanner Module (see page 137)

6.1.3.1.1.8 FPScannerStartCapturing Function

Starts capturing fingerprint image from certain device.

C++

```
NResult N_API FPScannerStartCapturing(HFPScanner hScanner);
```

Parameters

Parameters	Description
HFPScanner hScanner	[in] Handle to the FPScanner (see page 137) object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	hScanner (see page 150) or pValue is NULL (see page 133).
N_E_FAILED (see page 79)	Unspecified error has occurred.
N_E_INVALID_OPERATION (see page 80)	The scanner is already capturing.
N_E_NOT_REGISTERED	Module is not registered.

Module

FPScanner Module (see page 137)

6.1.3.1.1.9 FPScannerStartCapturingForOneImage Function

Starts capturing one fingerprint image from certain device.

C++

```
NResult N_API FPScannerStartCapturingForOneImage(HFPScanner hScanner);
```

Parameters

Parameters	Description
HFPScanner hScanner	[in] Handle to the FPScanner (see page 137) object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	hScanner (see page 150) or pValue is NULL (see page 133).
N_E_FAILED (see page 79)	Unspecified error has occurred.
N_E_INVALID_OPERATION (see page 80)	The scanner is already capturing.
N_E_NOT_REGISTERED	Module is not registered.

Module

FPScanner Module (see page 137)

6.1.3.1.1.10 FPScannerStopCapturing Function

Stops capturing fingerprint image.

C++

```
NResult N_API FPScannerStopCapturing(HFPScanner hScanner);
```

Parameters

Parameters	Description
HFPScanner hScanner	[in] Handle to the FPScanner (see page 137) object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	hScanner (see page 150) is NULL (see page 133).

Module

FPScanner Module (see page 137)

6.1.3.1.2 Variables**6.1.3.1.2.1 hImage Variable**

Handle to the image that has been scanned.

C++

```
HFPScanner HNGrayscaleImage hImage;
```

Module

FPScanner Module (see page 137)

6.1.3.1.3 Files**6.1.3.1.3.1 FPScanner.h**

Header file for FPScanner (see page 137) module.

Functions

	Name	Description
≡	FPScannerGetIdA (see page 137)	Gets associated device identifier.
≡	FPScannerGetIdW (see page 138)	Gets associated device identifier.
≡	FPScannerIsCapturing (see page 138)	Checks scanner status.
≡	FPScannerSetFingerPlacedCallback (see page 139)	Sets a callback function that is called immediately after finger is placed on a scanner.
≡	FPScannerSetFingerRemovedCallback (see page 139)	Sets a callback function that is called immediately after finger is removed from a scanner.
≡	FPScannerSetImageScannedCallback (see page 140)	Sets a callback function that is called immediately after image is scanned.
≡	FPScannerSetIsCapturingChangedCallback (see page 140)	Sets a callback function that is called immediately capturing state is changed.
≡	FPScannerStartCapturing (see page 140)	Starts capturing fingerprint image from certain device.
≡	FPScannerStartCapturingForOneImage (see page 141)	Starts capturing one fingerprint image from certain device.
≡	FPScannerStopCapturing (see page 141)	Stops capturing fingerprint image.

Module

FPScanner Module (see page 137)

Variables

Name	Description
hImage (see page 142)	Handle to the image that has been scanned.

6.1.3.2 FPScannerMan Module

Scanners manager enumerates, creates scanners.

Files

Name	Description
FPScannerMan.h (see page 150)	Header file for FPScannerMan. Scanners manager enumerates, creates scanners.

Functions

	Name	Description
≡	FPScannerManGetAvailableModulesA (see page 144)	Gets available fingerprint scanner modules.
≡	FPScannerManGetAvailableModulesW (see page 144)	Gets available fingerprint scanner modules.
≡	FPScannerManGetScanner (see page 144)	Retrieves the scanner at the specified index.
≡	FPScannerManGetScannerByIdA (see page 145)	Retrieves the scanner by specified identifier.
≡	FPScannerManGetScannerByIdW (see page 145)	Retrieves the scanner by specified identifier.
≡	FPScannerManGetScannerCount (see page 146)	Retrieves the number of scanners.
≡	FPScannerManInitialize (see page 146)	Initializes library.
≡	FPScannerManInitializeWithModulesA (see page 147)	Initializes library with concrete fingerprint scanners modules.
≡	FPScannerManInitializeWithModulesW (see page 147)	Initializes library with concrete fingerprint scanners modules.
≡	FPScannerManSetScannerAddedCallbackA (see page 148)	Sets a callback function that is called immediately after scanner is added.
≡	FPScannerManSetScannerAddedCallbackW (see page 148)	Sets a callback function that is called immediately after scanner is added.
≡	FPScannerManSetScannerRemovedCallbackA (see page 149)	Sets a callback function that is called immediately after scanner is removed.
≡	FPScannerManSetScannerRemovedCallbackW (see page 149)	Sets a callback function that is called immediately after scanner is removed.
≡	FPScannerManUninitialize (see page 149)	Uninitializes library.

Variables

Name	Description
hScanner (see page 150)	Handle to the scanner.
void * pParam (see page 150)	Parameter that was passed during callback registration.

6.1.3.2.1 Functions

6.1.3.2.1.1 FPScannerManGetAvailableModulesA Function

Gets available fingerprint scanner modules.

C++

```
NResult N_API FPScannerManGetAvailableModulesA(NChar * * pSzValue);
```

Parameters

Parameters	Description
NChar * * pSzValue	[out] Pointer to a list that contains available scanner modules.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

If the function fails, an error is returned.

Remarks

This function is used if compiling for platform without ANSI versions of the functions support.

If you can't choose between FPScannerManGetAvailableModulesA and FPScannerManGetAvailableModulesW ([see page 144](#)) functions, you can use FPScannerManGetAvailableModules.

Module

FPScannerMan Module ([see page 143](#))

6.1.3.2.1.2 FPScannerManGetAvailableModulesW Function

Gets available fingerprint scanner modules.

C++

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

Parameters

Parameters	Description
NWChar * * pSzValue	[out] Pointer to a list that contains available scanner modules.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

If the function fails, an error is returned.

Remarks

This function is used if compiling for platforms without Unicode support.

If you can't choose between FPScannerManGetAvailableModulesA ([see page 144](#)) and FPScannerManGetAvailableModulesW functions, you can use FPScannerManGetAvailableModules.

Module

FPScannerMan Module ([see page 143](#))

6.1.3.2.1.3 FPScannerManGetScanner Function

Retrieves the scanner at the specified index.

C++

```
NResult N_API FPScannerManGetScanner(NInt index, HFPScanner * pValue);
```

Parameters

Parameters	Description
NInt index	[in] Index of scanner to retrieve.
HFPScanner * pValue	[out] Points to HFPScanner that receives handle to FPSanner (see page 137).

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pValue</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Zero initialized FPSannerMan (see page 143) objects.

Module

FPSannerMan Module (see page 143)

6.1.3.2.1.4 FPSannerManGetScannerByIdA Function

Retrieves the scanner by specified identifier.

C++

```
NResult N_API FPSannerManGetScannerByIdA(const NChar * szId, HFPScanner * pValue);
```

Parameters

Parameters	Description
const NChar * szId	[in] Points to string that specifies the scanner.
HFPScanner * pValue	[out] Points to HFPScanner that receives handle to FPSanner (see page 137).

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pValue</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Zero initialized FPSannerMan (see page 143) objects.

Remarks

This function is used if compiling for platform without ANSI versions of the functions support.

If you can't choose between FPSannerManGetScannerByIdA and FPSannerManGetScannerByIdW functions, you can use FPSannerManGetScannerById.

Module

FPSannerMan Module (see page 143)

6.1.3.2.1.5 FPSannerManGetScannerByIdW Function

Retrieves the scanner by specified identifier.

C++

```
NResult N_API FPScannerManGetScannerByIdW(const NWChar * szId, HFPScanner * pValue);
```

Parameters

Parameters	Description
const NWChar * szId	[in] Points to string that specifies the scanner.
HFPScanner * pValue	[out] Points to HFPScanner that receives handle to FPScanner (see page 137).

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pValue</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Zero initialized FPScannerMan (see page 143) objects.

Remarks

This function is used if compiling for platforms without Unicode support.

If you can't choose between FPScannerManGetScannerByIDA and FPScannerManGetScannerByIdW functions, you can use FPScannerManGetScannerById.

Module

FPScannerMan Module (see page 143)

6.1.3.2.1.6 FPScannerManGetScannerCount Function

Retrieves the number of scanners.

C++

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

Parameters

Parameters	Description
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of scanners.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pValue</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Zero initialized FPScannerMan (see page 143) objects.

Module

FPScannerMan Module (see page 143)

6.1.3.2.1.7 FPScannerManInitialize Function

Initializes library.

C++

```
NResult N_API FPScannerManInitialize();
```

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error	Condition
N_E_FAILED (see page 79)	Unspecified error has occurred.

Module

FPScannerMan Module ([see page 143](#))

6.1.3.2.1.8 FPScannerManInitializeWithModulesA Function

Initializes library with concrete fingerprint scanners modules.

C++

```
NResult N_API FPScannerManInitializeWithModulesA(const NChar * szModules);
```

Parameters

Parameters	Description
const NChar * szModules	Point to a list of fingerprint scanner modules that should be initialized.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error	Condition
N_E_FAILED (see page 79)	Unspecified error has occurred.

Remarks

This function is used if compiling for platform without ANSI versions of the functions support.

If you can't choose between FPScannerManInitializeWithModulesA and FPScannerManInitializeWithModulesW ([see page 147](#)) functions, you can use FPScannerManInitializeWithModules.

Module

FPScannerMan Module ([see page 143](#))

6.1.3.2.1.9 FPScannerManInitializeWithModulesW Function

Initializes library with concrete fingerprint scanners modules.

C++

```
NResult N_API FPScannerManInitializeWithModulesW(const NWChar * szModules);
```

Parameters

Parameters	Description
const NWChar * szModules	Point to a list of fingerprint scanner modules that should be initialized.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error	Condition
N_E_FAILED (see page 79)	Unspecified error has occurred.

Remarks

This function is used if compiling for platforms without Unicode support.

If you can't choose between `FPScannerManInitializeWithModulesA` (see page 147) and `FPScannerManInitializeWithModulesW` functions, you can use `FPScannerManInitializeWithModules`.

Module

FPScannerMan Module (see page 143)

6.1.3.2.1.10 FPScannerManSetScannerAddedCallbackA Function

Sets a callback function that is called immediately after scanner is added.

C++

```
NResult N_API FPScannerManSetScannerAddedCallbackA(FPScannerManScannerCallbackA pCallback,  
void * pParam);
```

Parameters

Parameters	Description
FPScannerManScannerCallbackA pCallback	Address of the function that is called after scanner is added.
void * pParam	Parameter that is passed to the callback function.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Remarks

This function is used if compiling for platform without ANSI versions of the functions support.

Module

FPScannerMan Module (see page 143)

6.1.3.2.1.11 FPScannerManSetScannerAddedCallbackW Function

Sets a callback function that is called immediately after scanner is added.

C++

```
NResult N_API FPScannerManSetScannerAddedCallbackW(FPScannerManScannerCallbackW pCallback,  
void * pParam);
```

Parameters

Parameters	Description
FPScannerManScannerCallbackW pCallback	Address of the function that is called after scanner is added.
void * pParam	Parameter that is passed to the callback function.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Remarks

This function is used if compiling for platforms without Unicode support.

Module

FPScannerMan Module (see page 143)

6.1.3.2.1.12 FPScannerManSetScannerRemovedCallbackA Function

Sets a callback function that is called immediately after scanner is removed.

C++

```
NResult N_API FPScannerManSetScannerRemovedCallbackA(FPScannerManScannerCallbackA  
pCallback, void * pParam);
```

Parameters

Parameters	Description
FPScannerManScannerCallbackA pCallback	Address of the function that is called after scanner is removed.
void * pParam	Parameter that is passed to the callback function.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Remarks

This function is used if compiling for platform without ANSI versions of the functions support.

Module

FPScannerMan Module (see page 143)

6.1.3.2.1.13 FPScannerManSetScannerRemovedCallbackW Function

Sets a callback function that is called immediately after scanner is removed.

C++

```
NResult N_API FPScannerManSetScannerRemovedCallbackW(FPScannerManScannerCallbackW  
pCallback, void * pParam);
```

Parameters

Parameters	Description
FPScannerManScannerCallbackW pCallback	Address of the function that is called after scanner is removed.
void * pParam	Parameter that is passed to the callback function.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Remarks

This function is used if compiling for platforms without Unicode support.

Module

FPScannerMan Module (see page 143)

6.1.3.2.1.14 FPScannerManUninitialize Function

Uninitializes library.

C++

```
void N_API FPScannerManUninitialize();
```

Module

FPScannerMan Module ([see page 143](#))

6.1.3.2.2 Variables**6.1.3.2.2.1 hScanner Variable**

Handle to the scanner.

C++

```
HFPScanner hScanner;
```

Module

FPScannerMan Module ([see page 143](#))

6.1.3.2.2.2 void * pParam) Variable

Parameter that was passed during callback registration.

C++

```
const NWChar void * pParam);
```

Module

FPScannerMan Module ([see page 143](#))

6.1.3.2.3 Files**6.1.3.2.3.1 FPScannerMan.h**

Header file for FPScannerMan ([see page 143](#)). Scanners manager enumerates, creates scanners.

Functions

	Name	Description
	FPScannerManGetAvailableModulesA (see page 144)	Gets available fingerprint scanner modules.
	FPScannerManGetAvailableModulesW (see page 144)	Gets available fingerprint scanner modules.
	FPScannerManGetScanner (see page 144)	Retrieves the scanner at the specified index.
	FPScannerManGetScannerByIdA (see page 145)	Retrieves the scanner by specified identifier.
	FPScannerManGetScannerByIdW (see page 145)	Retrieves the scanner by specified identifier.
	FPScannerManGetScannerCount (see page 146)	Retrieves the number of scanners.
	FPScannerManInitialize (see page 146)	Initializes library.
	FPScannerManInitializeWithModulesA (see page 147)	Initializes library with concrete fingerprint scanners modules.

	FPScannerManInitializeWithModulesW (see page 147)	Initializes library with concrete fingerprint scanners modules.
	FPScannerManSetScannerAddedCallbackA (see page 148)	Sets a callback function that is called immediately after scanner is added.
	FPScannerManSetScannerAddedCallbackW (see page 148)	Sets a callback function that is called immediately after scanner is added.
	FPScannerManSetScannerRemovedCallbackA (see page 149)	Sets a callback function that is called immediately after scanner is removed.
	FPScannerManSetScannerRemovedCallbackW (see page 149)	Sets a callback function that is called immediately after scanner is removed.
	FPScannerManUninitialize (see page 149)	Uninitializes library.

Module

FPScannerMan Module ([see page 143](#))

Variables

Name	Description
hScanner (see page 150)	Handle to the scanner.
void * pParam) (see page 150)	Parameter that was passed during callback registration.

6.1.3.3 NDeviceManager Module

Provides functions for retrieving information about NDeviceManager ([see page 136](#)) library.

Files

Name	Description
NDeviceManager.h (see page 152)	Header file for the NDeviceManager (see page 136). Provides functions for retrieving information about NDeviceManager (see page 136) library.

Functions

	Name	Description
	NDeviceManagerGetInfo (see page 151)	Retrieves information about the NDeviceManager (see page 136) library.

6.1.3.3.1 Functions

6.1.3.3.1.1 NDeviceManagerGetInfo Function

Retrieves information about the NDeviceManager ([see page 136](#)) library.

C++

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

Parameters

Parameters	Description
NLibraryInfo * pValue	[out] A pointer to NLibraryInfo that contains different information about the library.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Module

NDeviceManager Module ([see page 151](#))

6.1.3.3.2 Files

6.1.3.3.2.1 NDeviceManager.h

Header file for the NDeviceManager ([see page 136](#)). Provides functions for retrieving information about NDeviceManager ([see page 136](#)) library.

Functions

	Name	Description
	NDeviceManagerGetInfo (see page 151)	Retrieves information about the NDeviceManager (see page 136) library.

Module

NDeviceManager Module ([see page 151](#))

6.1.4 NExtractors Library

Provides functionality for extracting faces, fingerprints, palmprints and irises features into templates.

Remarks**Requirements (Windows)**

- **Import Library:** NExtractors.dll.lib
- **DLL:** NExtractors.dll
- **Required DLLs:**
 - NCore.dll ([see page 74](#)).
 - NImages.dll ([see page 188](#))

Requirements (Linux)

- **Shared object:** libNExtractors.so.
- **Required objects:**
 - libNCore.so.
 - linNImages.so.

Notes

MegaMatcher On Card 2.0 SDK Extractor has the following restrictions:

For fingerprints extractor (NFExtractor):

- Only small templates can be extracted
- Template quality is not calculated
- Binarized/Skeletonized image is not returned (see NfeReturnedImage ([see page 161](#)))
- Ridge count, singular points, minutia G and curvatures are not returned

For faces extractor (NLExtractor):

- Only small templates can be extracted

- Only single face in image can be detected
- Faces rotated more than 15 degrees are not detected

Modules

Name	Description
NFExtractor Module (see page 153)	Provides functionality for extracting Neurotechnology Finger or Palm Records from fingerprint images.
NExtractors Module (see page 170)	Provides functions for retrieving NExtractors Library information.
NLExtractor Module (see page 171)	NLExtractor is a part of NBiometrics library intended for human face finding in pictures and extracting their features into Neurotechnology Faces Template (NLTemplate).

6.1.4.1 NFExtractor Module

Provides functionality for extracting Neurotechnology Finger or Palm Records from fingerprint images.

Files

Name	Description
NFExtractor.h (see page 168)	Header file for the NFExtractor module. Provides functionality for extracting Neurotechnology Finger Records (NFRecords) from fingerprint images using algorithm encapsulated in Neurotechnology Fingerprint Features Extractor (NFExtractor) object.
NFExtractorParams.h (see page 169)	Header file for the NFExtractor module. Enumerates different NFExtractor parameters.

Functions

	Name	Description
⇒	NfeExtract (see page 155)	Extracts NFRecord from the fingerprint or palmprint image using the specified NFExtractor.
⇒	NfeIsTemplateSmall (see page 156)	Checks if specified record is small template.
⇒	NfeMakeSmallTemplate (see page 156)	Makes a small template from specified record.
⇒	NfeUpdateTemplate (see page 157)	Updates the packed NFRecord using the specified image and the specified NFExtractor.
⇒	NfeGeneralizeEx (see page 157)	Generalizes count features collections to single features collection.
⇒	NfeCreate (see page 158)	Creates a HNFExtractor (see page 161). Created object can be used for extracting features from fingerprint images.
⇒	NfeCreateEx (see page 158)	Creates a HNFExtractor (see page 161). Created object can be used for extracting features from both fingerprint and palmprint images.
⇒	NfeGeneralize (see page 159)	Generalizes count features collections to single features collection.
⇒	NfeIsPalm (see page 160)	Checks if the NFExtractor is a palmprint extractor.
⇒	NfeMakeLightTemplate (see page 160)	Makes template for FingerCell matcher.
	HNFExtractor (see page 161)	Handle to NFExtractor object.

Macros

Name	Description
NFE_MODE_AUTHENTEC_AFS2 (see page 164)	The mode for FingerLoc AF-S2 (USB) scanner.
NFE_MODE_AUTHENTEC_AES2501B (see page 164)	The mode for EntrePad AES2501B (USB) scanner.

NFE_MODE_AUTHENTEC_AES4000 (see page 164)	The mode for EntrePad AES4000 (USB) scanner.
NFE_MODE_ATMEL_FINGERCHIP (see page 164)	The mode for Atmel FingerChip sensor.
NFE_MODE_BIOMETRIKA_FX2000 (see page 164)	The mode for Biometrika FX2000 scanner.
NFE_MODE_BIOMETRIKA_FX3000 (see page 164)	The mode for Biometrika FX3000 scanner.
NFE_MODE_BMF_BLP100 (see page 164)	The mode for BMF BLP100 scanner.
NFE_MODE_CROSSMATCH_VERIFIER300 (see page 165)	The mode for CrossMatch Verifier 300 scanner.
NFE_MODE_DIGENT_IZZIX (see page 165)	The mode for Digent Izzix FD1000 scanner.
NFE_MODE_DIGITALPERSONA_UAREU (see page 165)	The mode for Digital Persona U.are.U scanner.
NFE_MODE_FUJITSU_MBF200 (see page 165)	The mode for Fujitsu MBF200 scanner.
NFE_MODE_FUTRONIC_FS80 (see page 165)	The mode for Futronic FS80 scanner.
NFE_MODE_GENERAL (see page 165)	The general mode.
NFE_MODE_IDENTICATORTECHNOLOGY_DF90 (see page 165)	The mode for Identicator Technology DF90 scanner.
NFEP_GENERALIZATION_MIN_MINUTIA_COUNT (see page 165)	Identifier specifying generalization minimal minutiae count.
NFE_MODE_IDENTIX_DFR2090 (see page 165)	The mode for Identix DFR 2090 scanner.
NFEP_MIN_MINUTIA_COUNT (see page 166)	Identifier specifying minimal minutiae count.
NFE_MODE_IDENTIX_TOUCHVIEW (see page 166)	The mode for Identix TouchView scanner.
NFE_MODE_KEYTRONIC_SECUREDESKTOP (see page 166)	The mode for Keytronic Secure Desktop scanner.
NFE_MODE_LIGHTTUNING_LTTC500 (see page 166)	The mode for LTT-C500 Fingerprint Sensor scanner.
NFE_MODE_NITGEN_FINGKEY_HAMSTER (see page 166)	The mode for NITGEN Fingkey Hamster scanners.
NFE_MODE_PRECISEBIOMETRICS_100CS (see page 166)	The mode for Precise Biometrics 100CS scanner.
NFE_MODE_SECUGEN_HAMSTER (see page 166)	The mode for SecuGen Hamster scanner.
NFE_MODE_STARTEK_FM200 (see page 166)	The mode for Startek FM200 scanner.
NFE_MODE_TACOMA_CMOS (see page 166)	The mode for Tacoma CMOS scanner.
NFE_MODE_UPEK_TOUCHCHIP (see page 167)	The mode for UPEK TouchChip scanners.
NFEP_EXTRACTED_RIDGE_COUNTS (see page 167)	Identifier specifying type of ridge counts extracted parameter of type N_TYPE_INT (see page 103). Parameter value can be one of the NFRidgeCountsType (see page 578) enumeration members.
NFEP_GENERALIZATION_MAXIMAL_ROTATION (see page 167)	Maximal rotation of two features collection to each other. Must be in range 0°..180°.
NFEP_GENERALIZATION_THRESHOLD (see page 167)	Has the same meaning for features generalization as NFMP_MATCHING_THRESHOLD parameter for features matching.
NFEP_MODE (see page 167)	Identifier specifying mode (parameter value set) parameter of type N_TYPE_UINT (see page 104). Parameter value can be one of the NFE_MODE_XXX.
NFEP_QUALITY_THRESHOLD (see page 167)	Identifier specifies image quality threshold.

NFEP_RETURNED_IMAGE (see page 167)	Identifier specifying kind of image returned after extraction parameter of type N_TYPE_INT (see page 103). Parameter value can be one of the NfeReturnedImage (see page 161) enumeration members.
NFEP_TEMPLATE_SIZE (see page 167)	Identifier specifying template size parameter. Parameter value can be one of the NfeTemplateSize (see page 162) enumeration members.
NFEP_USE_QUALITY (see page 168)	If identifier is set to true (parameter of type N_TYPE_BOOL (see page 102)), then functions NfeExtract (see page 155), NfeExtractFromImage, NfeExtractUnpacked, NfeExtractUnpackedFromImage determines the quality of the image. If the image quality is lower than NFEP_QUALITY_THRESHOLD (see page 167) the function do not returns template.
NFE_MODE_ETHENTICA (see page 168)	The mode for Ethentica scanner.
NFE_MODE_TESTECH_BIOI (see page 168)	The mode for Testech BIOI scanner.

Structs, Records, Enums

	Name	Description
	NfeExtractionStatus_ (see page 161)	Specifies fingerprint or palmprint features extraction status.
	NfeReturnedImage_ (see page 161)	Specifies kind of image returned after extraction using NFEExtractor.
	NfeTemplateSize_ (see page 162)	Specifies returned NFRecord size after extraction using NFEExtractor.
	NfeExtractionStatus (see page 161)	Specifies fingerprint or palmprint features extraction status.
	NfeReturnedImage (see page 161)	Specifies kind of image returned after extraction using NFEExtractor.
	NfeTemplateSize (see page 162)	Specifies returned NFRecord size after extraction using NFEExtractor.

6.1.4.1.1 Functions

6.1.4.1.1.1 NfeExtract Function

Extracts NFRecord from the fingerprint or palmprint image using the specified NFEExtractor.

C++

```
NResult N_API NfeExtract(HNFEExtractor hExtractor, HNGrayscaleImage hImage, NFPosition position, NFImpressionType impressionType, NfeExtractionStatus * pExtractionStatus, HNFRecord * pHRecord);
```

Parameters

Parameters	Description
HNFEExtractor hExtractor	[in] Handle to the NFEExtractor object.
HNGrayscaleImage hImage	[in] Handle to the image.
NFPosition position	[in] Specifies finger or palm position.
NFImpressionType impressionType	[in] Specifies fingerprint or palmprint impression type.
NfeExtractionStatus * pExtractionStatus	[out] One of the NfeExtractionStatus (see page 161) value.
HNFRecord * pHRecord	[out] Pointer to HNFRecord (see page 571) that receives handle to created NFRecord object containing extracted template.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> , <i>hImage</i> (see page 142), <i>pExtractionStatus</i> or <i>pHRecord</i> is NULL (see page 133).

Remarks

position and *impressionType* are written to extracted NFRecord. Function accepts images with 50-5000 ppi. If image resolution is not equal 500 ppi it will be converted to 500 ppi. Recommended image resolution is 500 ppi.

Module

NFExtractor Module ([see page 153](#))

6.1.4.1.1.2 NfeIsTemplateSmall Function

Checks if specified record is small template.

C++

```
NResult N_API NfeIsTemplateSmall(HNFRecord hRecord, NBool * pValue);
```

Parameters

Parameters	Description
HNfRecord hRecord	[in] Handle to NFRecord to check if it is small template.
NBool * pValue	[out] Boolean value indicating if template is small.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)). Otherwise error code is returned. NSucceeded ([see page 83](#)) or NFailed ([see page 82](#)) macros can be used to detect if the function succeeded or failed.

See Also

NfeTemplateSize ([see page 162](#))

Module

NFExtractor Module ([see page 153](#))

6.1.4.1.1.3 NfeMakeSmallTemplate Function

Makes a small template from specified record.

C++

```
NResult N_API NfeMakeSmallTemplate(HNFRecord hRecord, HNFRecord * pHSmallRecord);
```

Parameters

Parameters	Description
HNfRecord hRecord	[in] Handle to NFRecord object to convert it to small template.
HNfRecord * pHSmallRecord	[out] Small template.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)). Otherwise error code is returned. NSucceeded ([see page 83](#)) or NFailed ([see page 82](#)) macros can be used to detect if the function succeeded or failed.

See Also

NfeTemplateSize ([see page 162](#))

Module

NFExtractor Module ([see page 153](#))

6.1.4.1.1.4 NfeUpdateTemplate Function

Updates the packed NFRecord using the specified image and the specified NFExtractor.

C++

```
NResult N_API NfeUpdateTemplate(HNExtractor hExtractor, HNGrayscaleImage hImage, HNRecord hRecord);
```

Parameters

Parameters	Description
HNExtractor hExtractor	[in] Handle to the NFExtractor object.
HNGrayscaleImage hImage	[in] Handle to the image.
HNRecord hRecord	[in] Handle to the NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hExtractor, hImage (see page 142) or hRecord is NULL (see page 133).

Remarks

This function updates ridge counts and minutiae curvatures and Gs using the image provided (in case of fingerprint image).

Module

NFExtractor Module (see page 153)

6.1.4.1.1.5 NfeGeneralizeEx Function

Generalizes count features collections to single features collection.

C++

```
NResult N_API NfeGeneralizeEx(HNExtractor hExtractor, NInt recordCount, const HNRecord * arHRecords, NfeExtractionStatus * pStatus, NInt * pBaseTemplateIndex, HNRecord * pHRecord);
```

Parameters

Parameters	Description
HNExtractor hExtractor	[in] Handle to the NFExtractor object.
NInt recordCount	[in] The number of finger records.
const HNRecord * arHRecords	[in] Pointer to memory buffer that contains NFRecord templates.
NfeExtractionStatus * pStatus	[out] Pointer to enumeration containing extraction status value.
NInt * pBaseTemplateIndex	[out] Index of main generalization template.
HNRecord * pHRecord	[out] Pointer to HNRecord (see page 571) that receives handle to created NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	One of <i>arHRecords</i> template is NULL (see page 133).
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> , <i>arHRecords</i> , <i>pHRecord</i> or <i>pBaseTemplateIndex</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>recordCount</i> is less 3 or greater than 10.

Module

NFExtractor Module (see page 153)

6.1.4.1.1.6 NfeCreate Function

Creates a HNFExtractor (see page 161). Created object can be used for extracting features from fingerprint images.

C++

```
NResult N_API NfeCreate(HNFExtractor * pHExtractor);
```

Parameters

Parameters	Description
HNFExtractor * pHExtractor	[out] Pointer to HNFExtractor (see page 161) that receives handle to created HNFExtractor (see page 161) object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pHExtractor is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

Created object must be deleted using NObjectFree (see page 97) function.

Module

NFExtractor Module (see page 153)

6.1.4.1.1.7 NfeCreateEx Function

Creates a HNFExtractor (see page 161). Created object can be used for extracting features from both fingerprint and palmprint images.

C++

```
NResult N_API NfeCreateEx(NBool isPalm, HNFExtractor * pHExtractor);
```

Parameters

Parameters	Description
NBool isPalm	[in] Boolean value indicating HNFExtractor (see page 161) type. If set to NTrue (see page 133) extractor for palmprint images will be created. Otherwise, extractor for fingerprint images will be created.
HNFExtractor * pHExtractor	[out] Pointer to HNFExtractor (see page 161) that receives handle to created HNFExtractor (see page 161) object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pHExtractor is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

Created object must be deleted using NObjectFree (see page 97) function.

Notes

HNExtractor (see page 161) can be used only for extracting fingerprint images or only for extracting palmprint images (i.e. the same HNExtractor (see page 161) can not be used both for extracting fingerprint and palmprint images). If you need to check if the HNExtractor (see page 161) was created for extracting features from palmprint images, you can use NfIsPalm (see page 160) function.

Module

NFExtractor Module (see page 153)

6.1.4.1.1.8 NfeGeneralize Function

Generalizes count features collections to single features collection.

C++

```
NResult N_API NfeGeneralize(HNExtractor hExtractor, NInt recordCount, const HNRecord *
arHRecords, HNRecord * pHRecord, NInt * pBaseTemplateIndex);
```

Parameters

Parameters	Description
HNExtractor hExtractor	[in] Handle to the NFExtractor object.
NInt recordCount	[in] The number of finger records.
const HNRecord * arHRecords	[in] Pointer to memory buffer that contains NFRecord templates.
HNRecord * pHRecord	[out] Pointer to HNRecord (see page 571) that receives handle to created NFRecord object.
NInt * pBaseTemplateIndex	[out] Index of main generalization template.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	One of arHRecords template is NULL (see page 133).
N_E_ARGUMENT_NULL (see page 78)	hExtractor, arHRecords, pHRecord or pBaseTemplateIndex is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	recordCount is less 3 or greater than 10.

Notes

This function is obsolete. Use NfeGeneralizeEx (see page 157) instead.

Module

NFExtractor Module ([see page 153](#))

6.1.4.1.1.9 NfeIsPalm Function

Checks if the NFExtractor is a palmprint extractor.

C++

```
NResult N_API NfeIsPalm(HNFExtractor hExtractor, NBool * pValue);
```

Parameters

Parameters	Description
HNFExtractor hExtractor	[in] Handle to the NFExtractor object.
NBool * pValue	[out] Boolean value indicating type of the NFExtractor. If value NTrue (see page 133) than NFExtractor was created for extracting features from palmprint images.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> or <i>pValue</i> is NULL (see page 133).

Module

NFExtractor Module ([see page 153](#))

6.1.4.1.1.10 NfeMakeLightTemplate Function

Makes template for FingerCell matcher.

C++

```
NResult N_API NfeMakeLightTemplate(HNFRecord hRecord, NBool makeSmallTemplate, NBool preserveRidgeCounts, HNFRecord * pHLightRecord);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NBool makeSmallTemplate	[in] Determines template size. If parameter will be NTrue (see page 133) then method returns NfeTemplateSize.nfetsSmall NFRecord FingerCell template; otherwise, NfeTemplateSize.nfetsLarge NFRecord FingerCell template.
NBool preserveRidgeCounts	[in] Reserved. Should be NFalse (see page 133).
HNFRecord * pHLightRecord	[out] Pointer to HNFRecord (see page 571) that receives handle to converted NFRecord object containing template.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pHLightRecord</i> is NULL (see page 133).

Module

NFExtractor Module (see page 153)

6.1.4.1.1.11 HNFExtractor

Handle to NFExtractor object.

Module

NFExtractor Module (see page 153)

6.1.4.1.2 Structs, Records, Enums

6.1.4.1.2.1 NfeExtractionStatus Enumeration

Specifies fingerprint or palmprint features extraction status.

C++

```
typedef enum NfeExtractionStatus_ {  
    nfeesTemplateCreated = 1,  
    nfeesTooFewMinutiae = 90,  
    nfeesQualityCheckFailed = 100,  
    nfeesMatchingFailed = 200  
} NfeExtractionStatus;
```

Members

Members	Description
nfeesTemplateCreated = 1	Features from fingerprint or palmprint image were extracted successfully.
nfeesTooFewMinutiae = 90	Indicates that fingerprint contains too few minutiae to extract features.
nfeesQualityCheckFailed = 100	Fingerprint or palmprint image has low quality.
nfeesMatchingFailed = 200	Indicates that fingerprints or palmprints failed to match.

Module

NFExtractor Module (see page 153)

6.1.4.1.2.2 NfeReturnedImage Enumeration

Specifies kind of image returned after extraction using NFExtractor.

C++

```
typedef enum NfeReturnedImage_ {  
    nferiNone = 0,  
    nferiBinarized = 1,  
    nferiSkeletonized = 2  
} NfeReturnedImage;
```

Members

Members	Description
nferiNone = 0	The image used for extraction is left unchanged.

nferiBinarized = 1	Binarized (filtered) image is written to the image used for extraction.
nferiSkeletonized = 2	Skeletonized image is written to the image used for extraction.

Module

NFExtractor Module ([see page 153](#))

6.1.4.1.2.3 NfeTemplateSize Enumeration

Specifies returned NFRecord size after extraction using NFExtractor.

C++

```
typedef enum NfeTemplateSize_ {
    nfetsSmall = 0,
    nfetsLarge = 256
} NfeTemplateSize;
```

Members

Members	Description
nfetsSmall = 0	Indicates that template contains small NFRecords.
nfetsLarge = 256	Indicates that template contains large NFRecords.

Module

NFExtractor Module ([see page 153](#))

6.1.4.1.3 Macros

To get a parameter value, use NObjectGetParameter ([see page 97](#)) function. To set parameter value - NObjectSetParameter ([see page 99](#)) function.

The following values can be used as *parameterId* in these functions:

- NFEP_EXTRACTED_RIDGE_COUNTS ([see page 167](#))
- NFEP_GENERALIZATION_MAXIMAL_ROTATION ([see page 167](#))
- NFEP_GENERALIZATION_THRESHOLD ([see page 167](#))
- NFEP_MODE ([see page 167](#)), NFEP_QUALITY_THRESHOLD ([see page 167](#))
- NFEP_RETURNED_IMAGE ([see page 167](#))
- NFEP_TEMPLATE_SIZE ([see page 167](#))
- NFEP_USE_QUALITY ([see page 168](#)).

Macros

Name	Description
NFE_MODE_AUTHENTEC_AFS2 (see page 164)	The mode for FingerLoc AF-S2 (USB) scanner.
NFE_MODE_AUTHENTEC_AES2501B (see page 164)	The mode for EntrePad AES2501B (USB) scanner.
NFE_MODE_AUTHENTEC_AES4000 (see page 164)	The mode for EntrePad AES4000 (USB) scanner.
NFE_MODE_ATMEL_FINGERCHIP (see page 164)	The mode for Atmel FingerChip sensor.
NFE_MODE_BIOMETRIKA_FX2000 (see page 164)	The mode for Biometrika FX2000 scanner.

NFE_MODE_BIOMETRIKA_FX3000 (see page 164)	The mode for Biometrika FX3000 scanner.
NFE_MODE_BMF_BLP100 (see page 164)	The mode for BMF BLP100 scanner.
NFE_MODE_CROSSMATCH_VERIFIER300 (see page 165)	The mode for CrossMatch Verifier 300 scanner.
NFE_MODE_DIGENT_IZZIX (see page 165)	The mode for Digent Izzix FD1000 scanner.
NFE_MODE_DIGITALPERSONA_UAREU (see page 165)	The mode for Digital Persona U.are.U scanner.
NFE_MODE_FUJITSU_MBF200 (see page 165)	The mode for Fujitsu MBF200 scanner.
NFE_MODE_FUTRONIC_FS80 (see page 165)	The mode for Futronic FS80 scanner.
NFE_MODE_GENERAL (see page 165)	The general mode.
NFE_MODE_IDENTICORTECHNOLOGY_DF90 (see page 165)	The mode for Identicator Technology DF90 scanner.
NFEP_GENERALIZATION_MIN_MINUTIA_COUNT (see page 165)	Identifier specifying generalization minimal minutiae count.
NFE_MODE_IDENTIX_DFR2090 (see page 165)	The mode for Identix DFR 2090 scanner.
NFEP_MIN_MINUTIA_COUNT (see page 166)	Identifier specifying minimal minutiae count.
NFE_MODE_IDENTIX_TOUCHVIEW (see page 166)	The mode for Identix TouchView scanner.
NFE_MODE_KEYTRONIC_SECUREDESKTOP (see page 166)	The mode for Keytronic Secure Desktop scanner.
NFE_MODE_LIGHTTUNING_LTTC500 (see page 166)	The mode for LTT-C500 Fingerprint Sensor scanner.
NFE_MODE_NITGEN_FINGKEY_HAMSTER (see page 166)	The mode for NITGEN Fingkey Hamster scanners.
NFE_MODE_PRECISEBIOMETRICS_100CS (see page 166)	The mode for Precise Biometrics 100CS scanner.
NFE_MODE_SECUGEN_HAMSTER (see page 166)	The mode for SecuGen Hamster scanner.
NFE_MODE_STARTEK_FM200 (see page 166)	The mode for Startek FM200 scanner.
NFE_MODE_TACOMA_CMOS (see page 166)	The mode for Tacoma CMOS scanner.
NFE_MODE_UPEK_TOUCHCHIP (see page 167)	The mode for UPEK TouchChip scanners.
NFEP_EXTRACTED_RIDGE_COUNTS (see page 167)	Identifier specifying type of ridge counts extracted parameter of type N_TYPE_INT (see page 103). Parameter value can be one of the NFRidgeCountsType (see page 578) enumeration members.
NFEP_GENERALIZATION_MAXIMAL_ROTATION (see page 167)	Maximal rotation of two features collection to each other. Must be in range 0°..180°.
NFEP_GENERALIZATION_THRESHOLD (see page 167)	Has the same meaning for features generalization as NFMP_MATCHING_THRESHOLD parameter for features matching.
NFEP_MODE (see page 167)	Identifier specifying mode (parameter value set) parameter of type N_TYPE_UINT (see page 104). Parameter value can be one of the NFE_MODE_XXX.
NFEP_QUALITY_THRESHOLD (see page 167)	Identifier specifies image quality threshold.
NFEP_RETURNED_IMAGE (see page 167)	Identifier specifying kind of image returned after extraction parameter of type N_TYPE_INT (see page 103). Parameter value can be one of the NfeReturnedImage (see page 161) enumeration members.
NFEP_TEMPLATE_SIZE (see page 167)	Identifier specifying template size parameter. Parameter value can be one of the NfeTemplateSize (see page 162) enumeration members.

NFEP_USE_QUALITY (see page 168)	If identifier is set to true (parameter of type N_TYPE_BOOL (see page 102)), then functions NfeExtract (see page 155), NfeExtractFromImage, NfeExtractUnpacked, NfeExtractUnpackedFromImage determines the quality of the image. If the image quality is lower than NFEP_QUALITY_THRESHOLD (see page 167) the function do not returns template.
NFE_MODE_ETHENTICA (see page 168)	The mode for Ethentica scanner.
NFE_MODE_TESTECH_BIOI (see page 168)	The mode for Testech BIOI scanner.

Module

NFExtractor Module (see page 153)

6.1.4.1.3.1 NFE_MODE_AUTHENTEC_AFS2 Macro

The mode for FingerLoc AF-S2 (USB) scanner.

C++

```
#define NFE_MODE_AUTHENTEC_AFS2 800
```

6.1.4.1.3.2 NFE_MODE_AUTHENTEC_AES2501B Macro

The mode for EntrePad AES2501B (USB) scanner.

C++

```
#define NFE_MODE_AUTHENTEC_AES2501B 811
```

6.1.4.1.3.3 NFE_MODE_AUTHENTEC_AES4000 Macro

The mode for EntrePad AES4000 (USB) scanner.

C++

```
#define NFE_MODE_AUTHENTEC_AES4000 810
```

6.1.4.1.3.4 NFE_MODE_ATMEL_FINGERCHIP Macro

The mode for Atmel FingerChip sensor.

C++

```
#define NFE_MODE_ATMEL_FINGERCHIP 900
```

6.1.4.1.3.5 NFE_MODE_BIOMETRIKA_FX2000 Macro

The mode for Biometrika FX2000 scanner.

C++

```
#define NFE_MODE_BIOMETRIKA_FX2000 200
```

6.1.4.1.3.6 NFE_MODE_BIOMETRIKA_FX3000 Macro

The mode for Biometrika FX3000 scanner.

C++

```
#define NFE_MODE_BIOMETRIKA_FX3000 201
```

6.1.4.1.3.7 NFE_MODE_BMF_BLP100 Macro

The mode for BMF BLP100 scanner.

C++

```
#define NFE_MODE_BMF_BLP100 1000
```

6.1.4.1.3.8 NFE_MODE_CROSSMATCH_VERIFIER300 Macro

The mode for CrossMatch Verifier 300 scanner.

C++

```
#define NFE_MODE_CROSSMATCH_VERIFIER300 1300
```

6.1.4.1.3.9 NFE_MODE_DIGENT_IZZIX Macro

The mode for Digent Izzix FD1000 scanner.

C++

```
#define NFE_MODE_DIGENT_IZZIX 1600
```

6.1.4.1.3.10 NFE_MODE_DIGITALPERSONA_UAREU Macro

The mode for Digital Persona U.are.U scanner.

C++

```
#define NFE_MODE_DIGITALPERSONA_UAREU 100
```

6.1.4.1.3.11 NFE_MODE_FUJITSU_MBF200 Macro

The mode for Fujitsu MBF200 scanner.

C++

```
#define NFE_MODE_FUJITSU_MBF200 1800
```

6.1.4.1.3.12 NFE_MODE_FUTRONIC_FS80 Macro

The mode for Futronic FS80 scanner.

C++

```
#define NFE_MODE_FUTRONIC_FS80 1900
```

6.1.4.1.3.13 NFE_MODE_GENERAL Macro

The general mode.

C++

```
#define NFE_MODE_GENERAL 0
```

6.1.4.1.3.14 NFE_MODE_IDENTICATORTECHNOLOGY_DF90 Macro

The mode for Identicator Technology DF90 scanner.

C++

```
#define NFE_MODE_IDENTICATORTECHNOLOGY_DF90 700
```

6.1.4.1.3.15 NFEP_GENERALIZATION_MIN_MINUTIA_COUNT Macro

Identifier specifying generalization minimal minutiae count.

C++

```
#define NFEP_GENERALIZATION_MIN_MINUTIA_COUNT 319
```

6.1.4.1.3.16 NFE_MODE_IDENTIX_DFR2090 Macro

The mode for Identix DFR 2090 scanner.

C++

```
#define NFE_MODE_IDENTIX_DFR2090 401
```

6.1.4.1.3.17 NFE_MIN_MINUTIA_COUNT Macro

Identifier specifying minimal minutiae count.

C++

```
#define NFE_MIN_MINUTIA_COUNT 119
```

6.1.4.1.3.18 NFE_MODE_IDENTIX_TOUCHVIEW Macro

The mode for Identix TouchView scanner.

C++

```
#define NFE_MODE_IDENTIX_TOUCHVIEW 400
```

6.1.4.1.3.19 NFE_MODE_KEYTRONIC_SECUREDESKTOP Macro

The mode for Keytronic Secure Desktop scanner.

C++

```
#define NFE_MODE_KEYTRONIC_SECUREDESKTOP 300
```

6.1.4.1.3.20 NFE_MODE_LIGHTUNING_LTTC500 Macro

The mode for LTT-C500 Fingerprint Sensor scanner.

C++

```
#define NFE_MODE_LIGHTUNING_LTTC500 2000
```

6.1.4.1.3.21 NFE_MODE_NITGEN_FINGKEY_HAMSTER Macro

The mode for NITGEN Fingkey Hamster scanners.

C++

```
#define NFE_MODE_NITGEN_FINGKEY_HAMSTER 1400
```

6.1.4.1.3.22 NFE_MODE_PRECISEBIOMETRICS_100CS Macro

The mode for Precise Biometrics 100CS scanner.

C++

```
#define NFE_MODE_PRECISEBIOMETRICS_100CS 500
```

6.1.4.1.3.23 NFE_MODE_SECUGEN_HAMSTER Macro

The mode for SecuGen Hamster scanner.

C++

```
#define NFE_MODE_SECUGEN_HAMSTER 1100
```

6.1.4.1.3.24 NFE_MODE_STARTEK_FM200 Macro

The mode for Startek FM200 scanner.

C++

```
#define NFE_MODE_STARTEK_FM200 1700
```

6.1.4.1.3.25 NFE_MODE_TACOMA_CMOS Macro

The mode for Tacoma CMOS scanner.

C++

```
#define NFE_MODE_TACOMA_CMOS 2100
```

6.1.4.1.3.26 NFE_MODE_UPEK_TOUCHCHIP Macro

The mode for UPEK TouchChip scanners.

C++

```
#define NFE_MODE_UPEK_TOUCHCHIP 600
```

6.1.4.1.3.27 NFEP_EXTRACTED_RIDGE_COUNTS Macro

Identifier specifying type of ridge counts extracted parameter of type N_TYPE_INT (see page 103). Parameter value can be one of the NFRidgeCountsType (see page 578) enumeration members.

C++

```
#define NFEP_EXTRACTED_RIDGE_COUNTS 120
```

6.1.4.1.3.28 NFEP_GENERALIZATION_MAXIMAL_ROTATION Macro

Maximal rotation of two features collection to each other. Must be in range 0°..180°.

C++

```
#define NFEP_GENERALIZATION_MAXIMAL_ROTATION 301
```

6.1.4.1.3.29 NFEP_GENERALIZATION_THRESHOLD Macro

Has the same meaning for features generalization as NFMP_MATCHING_THRESHOLD parameter for features matching.

C++

```
#define NFEP_GENERALIZATION_THRESHOLD 300
```

6.1.4.1.3.30 NFEP_MODE Macro

Identifier specifying mode (parameter value set) parameter of type N_TYPE_UINT (see page 104). Parameter value can be one of the NFE_MODE_XXX.

C++

```
#define NFEP_MODE 1000
```

6.1.4.1.3.31 NFEP_QUALITY_THRESHOLD Macro

Identifier specifies image quality threshold.

C++

```
#define NFEP_QUALITY_THRESHOLD 910
```

6.1.4.1.3.32 NFEP_RETURNED_IMAGE Macro

Identifier specifying kind of image returned after extraction parameter of type N_TYPE_INT (see page 103). Parameter value can be one of the NfeReturnedImage (see page 161) enumeration members.

C++

```
#define NFEP_RETURNED_IMAGE 190
```

6.1.4.1.3.33 NFEP_TEMPLATE_SIZE Macro

Identifier specifying template size parameter. Parameter value can be one of the NfeTemplateSize (see page 162) enumeration members.

C++

```
#define NFEP_TEMPLATE_SIZE 100
```

6.1.4.1.3.34 NFEP_USE_QUALITY Macro

If identifier is set to true (parameter of type N_TYPE_BOOL ([see page 102](#))), then functions NfeExtract ([see page 155](#)), NfeExtractFromImage, NfeExtractUnpacked, NfeExtractUnpackedFromImage determines the quality of the image. If the image quality is lower than NFEP_QUALITY_THRESHOLD ([see page 167](#)) the function does not return template.

C++

```
#define NFEP_USE_QUALITY 900
```

6.1.4.1.3.35 NFE_MODE_ETHENTICA Macro

The mode for Ethentica scanner.

C++

```
#define NFE_MODE_ETHENTICA 1200
```

6.1.4.1.3.36 NFE_MODE_TESTECH_BIOI Macro

The mode for Testech BIOI scanner.

C++

```
#define NFE_MODE_TESTECH_BIOI 1500
```

6.1.4.1.4 Files

6.1.4.1.4.1 NFExtractor.h

Header file for the NFExtractor module. Provides functionality for extracting Neurotechnology Finger Records (NFRecords) from fingerprint images using algorithm encapsulated in Neurotechnology Fingerprint Features Extractor (NFExtractor) object.

Enumerations

	Name	Description
	NfeExtractionStatus_ (see page 161)	Specifies fingerprint or palmprint features extraction status.
	NfeExtractionStatus (see page 161)	Specifies fingerprint or palmprint features extraction status.

Functions

	Name	Description
	NfeCreate (see page 158)	Creates a HNFExtractor (see page 161). Created object can be used for extracting features from fingerprint images.
	NfeCreateEx (see page 158)	Creates a HNFExtractor (see page 161). Created object can be used for extracting features from both fingerprint and palmprint images.
	NfeExtract (see page 155)	Extracts NFRecord from the fingerprint or palmprint image using the specified NFExtractor.
	NfeGeneralize (see page 159)	Generalizes count features collections to single features collection.
	NfeGeneralizeEx (see page 157)	Generalizes count features collections to single features collection.
	NfeIsPalm (see page 160)	Checks if the NFExtractor is a palmprint extractor.
	NfeIsTemplateSmall (see page 156)	Checks if specified record is small template.

	NfeMakeLightTemplate (see page 160)	Makes template for FingerCell matcher.
	NfeMakeSmallTemplate (see page 156)	Makes a small template from specified record.
	NfeUpdateTemplate (see page 157)	Updates the packed NFRecord using the specified image and the specified NFEExtractor.

Module

NFEExtractor Module ([see page 153](#))

6.1.4.1.4.2 NFEExtractorParams.h

Header file for the NFEExtractor module. Enumerates different NFEExtractor parameters.

Enumerations

	Name	Description
	NfeReturnedImage_ (see page 161)	Specifies kind of image returned after extraction using NFEExtractor.
	NfeTemplateSize_ (see page 162)	Specifies returned NFRecord size after extraction using NFEExtractor.
	NfeReturnedImage (see page 161)	Specifies kind of image returned after extraction using NFEExtractor.
	NfeTemplateSize (see page 162)	Specifies returned NFRecord size after extraction using NFEExtractor.

Macros

Name	Description
NFE_MODE_ATMEL_FINGERCHIP (see page 164)	The mode for Atmel FingerChip sensor.
NFE_MODE_AUTHENTEC_AES2501B (see page 164)	The mode for EntrePad AES2501B (USB) scanner.
NFE_MODE_AUTHENTEC_AES4000 (see page 164)	The mode for EntrePad AES4000 (USB) scanner.
NFE_MODE_AUTHENTEC_AFS2 (see page 164)	The mode for FingerLoc AF-S2 (USB) scanner.
NFE_MODE_BIOMETRIKA_FX2000 (see page 164)	The mode for Biometrika FX2000 scanner.
NFE_MODE_BIOMETRIKA_FX3000 (see page 164)	The mode for Biometrika FX3000 scanner.
NFE_MODE_BMF_BLP100 (see page 164)	The mode for BMF BLP100 scanner.
NFE_MODE_CROSSMATCH_VERIFIER300 (see page 165)	The mode for CrossMatch Verifier 300 scanner.
NFE_MODE_DIGENT_IZZIX (see page 165)	The mode for Digent Izzix FD1000 scanner.
NFE_MODE_DIGITALPERSONA_UAREU (see page 165)	The mode for Digital Persona U.are.U scanner.
NFE_MODE_ETHENTICA (see page 168)	The mode for Ethentica scanner.
NFE_MODE_FUJITSU_MBF200 (see page 165)	The mode for Fujitsu MBF200 scanner.
NFE_MODE_FUTRONIC_FS80 (see page 165)	The mode for Futronic FS80 scanner.
NFE_MODE_GENERAL (see page 165)	The general mode.
NFE_MODE_IDENTICATORTECHNOLOGY_DF90 (see page 165)	The mode for Identicator Technology DF90 scanner.
NFE_MODE_IDENTIX_DFR2090 (see page 165)	The mode for Identix DFR 2090 scanner.

NFE_MODE_IDENTIX_TOUCHVIEW (see page 166)	The mode for Identix TouchView scanner.
NFE_MODE_KEYTRONIC_SECUREDESKTOP (see page 166)	The mode for Keytronic Secure Desktop scanner.
NFE_MODE_LIGHTTUNING_LTTC500 (see page 166)	The mode for LTT-C500 Fingerprint Sensor scanner.
NFE_MODE_NITGEN_FINGKEY_HAMSTER (see page 166)	The mode for NITGEN Fingkey Hamster scanners.
NFE_MODE_PRECISEBIOMETRICS_100CS (see page 166)	The mode for Precise Biometrics 100CS scanner.
NFE_MODE_SECUGEN_HAMSTER (see page 166)	The mode for SecuGen Hamster scanner.
NFE_MODE_STARTEK_FM200 (see page 166)	The mode for Startek FM200 scanner.
NFE_MODE_TACOMA_CMOS (see page 166)	The mode for Tacoma CMOS scanner.
NFE_MODE_TESTECH_BIOI (see page 168)	The mode for Testech BIOI scanner.
NFE_MODE_UPEK_TOUCHCHIP (see page 167)	The mode for UPEK TouchChip scanners.
NFEP_EXTRACTED_RIDGE_COUNTS (see page 167)	Identifier specifying type of ridge counts extracted parameter of type N_TYPE_INT (see page 103). Parameter value can be one of the NFRidgeCountsType (see page 578) enumeration members.
NFEP_GENERALIZATION_MAXIMAL_ROTATION (see page 167)	Maximal rotation of two features collection to each other. Must be in range 0°..180°.
NFEP_GENERALIZATION_MIN_MINUTIA_COUNT (see page 165)	Identifier specifying generalization minimal minutiae count.
NFEP_GENERALIZATION_THRESHOLD (see page 167)	Has the same meaning for features generalization as NFMP_MATCHING_THRESHOLD parameter for features matching.
NFEP_MIN_MINUTIA_COUNT (see page 166)	Identifier specifying minimal minutiae count.
NFEP_MODE (see page 167)	Identifier specifying mode (parameter value set) parameter of type N_TYPE_UINT (see page 104). Parameter value can be one of the NFE_MODE_XXX.
NFEP_QUALITY_THRESHOLD (see page 167)	Identifier specifies image quality threshold.
NFEP_RETURNED_IMAGE (see page 167)	Identifier specifying kind of image returned after extraction parameter of type N_TYPE_INT (see page 103). Parameter value can be one of the NfeReturnedImage (see page 161) enumeration members.
NFEP_TEMPLATE_SIZE (see page 167)	Identifier specifying template size parameter. Parameter value can be one of the NfeTemplateSize (see page 162) enumeration members.
NFEP_USE_QUALITY (see page 168)	If identifier is set to true (parameter of type N_TYPE_BOOL (see page 102)), then functions NfeExtract (see page 155), NfeExtractFromImage, NfeExtractUnpacked, NfeExtractUnpackedFromImage determines the quality of the image. If the image quality is lower than NFEP_QUALITY_THRESHOLD (see page 167) the function do not returns template.

Module

NFExtractor Module ([see page 153](#))

6.1.4.2 NExtractors Module

Provides functions for retrieving NExtractors Library information.

Files

Name	Description
NExtractors.h (see page 171)	Header file for the NExtractor. Provides functions for retrieving information about NExtractors library.

Functions

	Name	Description
	NExtractorsGetInfo (see page 171)	Retrieves information about the NExtractors library.

6.1.4.2.1 Functions

6.1.4.2.1.1 NExtractorsGetInfo Function

Retrieves information about the NExtractors library.

C++

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

Parameters

Parameters	Description
NLibraryInfo * pValue	[out] A pointer to NLibraryInfo that contains different information about the library.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Module

NExtractors Module ([see page 170](#))

6.1.4.2.2 Files

6.1.4.2.2.1 NExtractors.h

Header file for the NExtractor. Provides functions for retrieving information about NExtractors library.

Functions

	Name	Description
	NExtractorsGetInfo (see page 171)	Retrieves information about the NExtractors library.

Module

NExtractors Module ([see page 170](#))

6.1.4.3 NLExtractor Module

NLExtractor is a part of NBiometrics library intended for human face finding in pictures and extracting their features into Neurotechnology Faces Template (NLTemplate).

Files

Name	Description
NLExtractor.h (see page 186)	Header file for the NLExtractor library. NLExtractor is a part of NLExtractor library intended for human face finding in pictures and extracting their features into Neurotechnology Faces Template(NLTemplate).
NLExtractorParams.h (see page 187)	C header file which contains NLExtractor parameters.

Functions

	Name	Description
⇒	NleCompressEx (see page 173)	Compresses the features extracted or generalized by the extraction routines to be stored or used in matching.
⇒	NleDetectFace (see page 174)	Finds one face region that best fits extraction requirements from the given image.
⇒	NleDetectFaces (see page 174)	Finds all face regions in the given image.
⇒	NleDetectFacialFeatures (see page 175)	Finds locations of facial feature points.
⇒	NleExtract (see page 175)	Fully automatically extracts feature template from the face that best fits extraction requirements from the given image.
⇒	NleExtractNext (see page 176)	Extracts one features template from image sequence
⇒	NleExtractNextEx (see page 177)	Extracts one features template from image sequence.
⇒	NleExtractUsingDetails (see page 177)	Extracts feature template from the face with given detection details from an image.
⇒	NleGeneralizeEx (see page 178)	Generates generalized template from uncompressed set of templates.
⇒	NleCompress (see page 178)	Compresses the features extracted or generalized by the extraction routines to be stored or used in matching.
⇒	NleCreate (see page 179)	Creates a NLExtractor.
⇒	NleExtractStart (see page 179)	Initializes features extraction routines that use image sequences to extract features.
⇒	NleGeneralize (see page 180)	Generates generalized template from uncompressed set of templates.
	HNLExtractor (see page 180)	Handle to NLExtractor object.

Macros

Name	Description
NLEP_FACE_CONFIDENCE_THRESHOLD (see page 184)	Identifier of type N_TYPE_DOUBLE (see page 103). Specifies the threshold which is considered when looking for faces in an image. For each face candidate confidence parameter is calculated. With higher threshold value faces are selected more strictly by the face detection routines. Must be in range [0..100].
NLEP_MAX_ROLL_ANGLE_DEVIATION (see page 184)	Defines maximum roll angle deviation from frontal face in degrees which is considered when looking for faces in an image. Must be in range [0..180].
NLEP_TEMPLATE_SIZE (see page 185)	Defines template size of Face template (NLTemplate).
NLEP_FACE_QUALITY_THRESHOLD (see page 185)	Identifier of type N_TYPE_BYTE (see page 102). Specifies the threshold which is considered when extracting facial features from the image. With higher threshold better quality of face image is required to successfully extract facial features. The value of this parameter can be in range [0..255]. The default value is 128.
NLEP_FAVOR_LARGEST_FACE (see page 185)	Identifier of type N_TYPE_BOOL (see page 102). If set to true the largest face found in an image will be used for facial feature extraction instead of the one with the highest face confidence threshold score.

NLEP_LIVENESS_THRESHOLD (see page 185)	Identifier of type N_TYPE_DOUBLE (see page 103). This threshold checks if faces are extracted from a live image stream. The higher the value of this parameter, the more strictly the stream is checked if the face in it is real (for example if it's not forged by showing a photo of a person in front a camera). The value of this parameter can be in range [0..100]. The default value is 50.
NLEP_MAX_IOD (see page 185)	Identifier of type N_TYPE_INT (see page 103) specifying maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].
NLEP_MAX_RECORDS_PER_TEMPLATE (see page 186)	Identifier of type N_TYPE_INT (see page 103). Sets the maximum number of records an extraction function can return in one NLTemplate. The value of this parameter can be in range [1..16]. The default value is 5.
NLEP_MIN_IOD (see page 186)	Identifier of type N_TYPE_INT (see page 103) specifying minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].
NLEP_USE_LIVENESS_CHECK (see page 186)	Identifier of type N_TYPE_BOOL (see page 102). If set to true liveness check is performed while doing facial feature extraction from an image stream.

Structs, Records, Enums

	Name	Description
	NleRotation_ (see page 181)	Enumerates face rotation values.
	NleTemplateSize_ (see page 181)	Enumerates available template size values.
	NleRotation (see page 181)	Enumerates face rotation values.
	NleTemplateSize (see page 181)	Enumerates available template size values.
	NleDetectionDetails_ (see page 182)	Structure with information about face detection results in a face detection routine.
	NleEyes_ (see page 182)	Structure defining information of an eye pair.
	NleExtractionStatus_ (see page 182)	Face features extraction status.
	NleFace_ (see page 183)	Structure describing face region information
	NleDetectionDetails (see page 182)	Structure with information about face detection results in a face detection routine.
	NleEyes (see page 182)	Structure defining information of an eye pair.
	NleExtractionStatus (see page 182)	Face features extraction status.
	NleFace (see page 183)	Structure describing face region information

6.1.4.3.1 Functions

6.1.4.3.1.1 NleCompressEx Function

Compresses the features extracted or generalized by the extraction routines to be stored or used in matching.

C++

```
NResult N_API NleCompressEx(HNLTemplate hSrcTemplate, NleTemplateSize dstTemplateSize,  
HNLTemplate * pHDstTemplate);
```

Parameters

Parameters	Description
HNLTemplate hSrcTemplate	[in] Handle to the NLTemplate object to be compressed.

NleTemplateSize dstTemplateSize	[in] Size of compressed template. Can be used small, medium or large template. For more information see NLEP_TEMPLATE_SIZE (see page 185) parameter.
HNLTemplate * pHDstTemplate	[out] Pointer to HNLTemplate that receives handle to created NLTemplate object.

Returns

If the function succeeds, the return value is N_OK (see page 82) or NSucceeded (see page 83). Otherwise, error code is returned.

Module

NLExtractor Module (see page 171)

6.1.4.3.1.2 NleDetectFace Function

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

C++

```
NResult N_API NleDetectFace(HNLExtractor hExtractor, HNGrayscaleImage hImage, NBool * pDetected, NleFace * pFace);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle of the NLExtractor object.
HNGrayscaleImage hImage	[in] Handle of the source image.
NBool * pDetected	[out] Pointer to NBool (see page 116) variable that is set to true if face is found in the image else it's false
NleFace * pFace	[out] Pointer to the NleFace (see page 183) structure of found face.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hExtractor, hImage (see page 142), pDetected or pFace is NULL (see page 133).

Module

NLExtractor Module (see page 171)

6.1.4.3.1.3 NleDetectFaces Function

Finds all face regions in the given image.

C++

```
NResult N_API NleDetectFaces(HNLExtractor hExtractor, HNGrayscaleImage hImage, NInt * pFaceCount, NleFace * * pArFaces);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
HNGrayscaleImage hImage	[in] Handle to the source image.
NInt * pFaceCount	[out] Pointer to the NInt (see page 117) variable that is set to the number of found faces in the image.

NleFace ** pArFaces	[out] pointer to an array of NleFace (see page 183) structures of found faces.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> , <i>hImage</i> (see page 142), <i>pFaceCount</i> or <i>pArFaces</i> is NULL (see page 133).

Remarks

The buffer for the *pArFaces* variable is allocated inside this function automatically, but you must free this variable yourself when it becomes unnecessary for you. It must be freed using NCore (see page 74) library function NFree (see page 94).

Module

NLExtractor Module (see page 171)

6.1.4.3.1.4 NleDetectFacialFeatures Function

Finds locations of facial feature points.

C++

```
NResult N_API NleDetectFacialFeatures(HNLExtractor hExtractor, HNGrayscaleImage hImage,
const NleFace * pFace, NleDetectionDetails * pDetectionDetails);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
HNGrayscaleImage hImage	[in] Handle to the source image.
const NleFace * pFace	[out] Pointer to the NleFace (see page 183) structure of a face found in the image hImage (see page 142).
NleDetectionDetails * pDetectionDetails	[out] Pointer to the NleDetectionDetails (see page 182) structure of face detection results in the image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> , <i>hImage</i> (see page 142), <i>pFace</i> or <i>pDetectionDetails</i> is NULL (see page 133).

Module

NLExtractor Module (see page 171)

6.1.4.3.1.5 NleExtract Function

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

C++

```
NResult N_API NleExtract(HNLExtractor hExtractor, HNGrayscaleImage hImage,
NleDetectionDetails * pDetectionDetails, NleExtractionStatus * pStatus, HNLTemplate *
pHTemplate);
```


Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
HNGrayscaleImage hImage	[in] Handle to the source image.
NleDetectionDetails * pDetectionDetails	[out] Pointer to the NleDetectionDetails (see page 182) structure of face detection results in the image.
NleExtractionStatus * pStatus	[out] Pointer to NleExtractionStatus (see page 182) that receives extraction status.
HNLTemplate * pHTemplate	[out] Pointer to HNLTemplate that receives handle to created NLTemplate object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> , <i>hImage</i> (see page 142), <i>pDetectionDetails</i> , <i>pStatus</i> or <i>pHTemplate</i> is NULL (see page 133).

Module

NLExtractor Module (see page 171)

6.1.4.3.1.6 NleExtractNext Function

Extracts one features template from image sequence

C++

```
NResult N_API NleExtractNext(HNLExtractor hExtractor, HNGrayscaleImage hImage,
NleDetectionDetails * pDetectionDetails, HNLTemplate * pHTemplate, NleExtractionStatus *
pStatus);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
HNGrayscaleImage hImage	[in] Handle to the source image.
NleDetectionDetails * pDetectionDetails	[out] Pointer to the NleDetectionDetails (see page 182) structure of face detection results in the image.
HNLTemplate * pHTemplate	[out] Pointer to HNLTemplate that receives handle to created NLTemplate object.
NleExtractionStatus * pStatus	[out] Pointer to NleExtractionStatus (see page 182) that receives extraction status.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> , <i>hImage</i> (see page 142), <i>pDetectionDetails</i> , <i>pStatus</i> or <i>pHTemplate</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	NleExtractStart (see page 179) function was not called before NleExtractNext.

Notes

This function is obsolete. Use `NleExtractNextEx` (see page 177) instead.

Module

NLExtractor Module (see page 171)

6.1.4.3.1.7 NleExtractNextEx Function

Extracts one features template from image sequence.

C++

```
NResult N_API NleExtractNextEx(HNLExtractor hExtractor, HNGrayscaleImage hImage,
NleDetectionDetails * pDetectionDetails, NInt * pBaseFrameIndex, HNLTemplate * pHTemplate,
NleExtractionStatus * pStatus);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
HNGrayscaleImage hImage	[in] Handle to the source image.
NleDetectionDetails * pDetectionDetails	[out] Pointer to the NleDetectionDetails (see page 182) structure of face detection results in the image.
NInt * pBaseFrameIndex	[out] ?Index of base frame.
HNLTemplate * pHTemplate	[out] Pointer to HNLTemplate that receives handle to created NLTTemplate object.
NleExtractionStatus * pStatus	[out] Pointer to NleExtractionStatus (see page 182) that receives extraction status.

Returns

If the function succeeds, the return value is `N_OK` (see page 82). Otherwise error code is returned. `NSucceeded` (see page 83) or `NFailed` (see page 82) macros can be used to detect if the function succeeded or failed.

Module

NLExtractor Module (see page 171)

6.1.4.3.1.8 NleExtractUsingDetails Function

Extracts feature template from the face with given detection details from an image.

C++

```
NResult N_API NleExtractUsingDetails(HNLExtractor hExtractor, HNGrayscaleImage hImage,
NleDetectionDetails * pDetectionDetails, NleExtractionStatus * pStatus, HNLTemplate *
pHTemplate);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
HNGrayscaleImage hImage	[in] Handle to the source image.
NleDetectionDetails * pDetectionDetails	[out] Pointer to the NleDetectionDetails (see page 182) structure of face detection results in the image.
NleExtractionStatus * pStatus	[out] Pointer to NleExtractionStatus (see page 182) that receives extraction status.
HNLTemplate * pHTemplate	[out] Pointer to HNLTemplate that receives handle to created NLTTemplate object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> , <i>hImage</i> (see page 142), <i>pDetectionDetails</i> , <i>pStatus</i> or <i>pHTemplate</i> is NULL (see page 133).

Module

NLExtractor Module (see page 171)

6.1.4.3.1.9 NleGeneralizeEx Function

Generates generalized template from uncompressed set of templates.

C++

```
NResult N_API NleGeneralizeEx(HNLExtractor hExtractor, NInt templateCount, const
HNLTemplate * arHTemplates, NleExtractionStatus * pStatus, NInt * pBaseTemplateIndex,
HNLTemplate * pHTemplate);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
NInt templateCount	[in] Count of templates to be used in generalization.
const HNLTemplate * arHTemplates	[out] Pointer to memory buffer that contains NLTTemplate templates.
NleExtractionStatus * pStatus	[out] Template extraction status.
NInt * pBaseTemplateIndex	[out] Index of main generalization template.
HNLTemplate * pHTemplate	[out] Pointer to HNLTemplate that receives handle to created NLTTemplate object.

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise error code is returned. NSucceeded (see page 83) or NFailed (see page 82) macros can be used to detect if the function succeeded or failed.

Module

NLExtractor Module (see page 171)

6.1.4.3.1.10 NleCompress Function

Compresses the features extracted or generalized by the extraction routines to be stored or used in matching.

C++

```
NResult N_API NleCompress(HNLExtractor hExtractor, HNLTemplate hTemplate, HNLTemplate *
pHCompressedTemplate);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
HNLTemplate hTemplate	[in] Handle to the NTemplate object.
HNLTemplate * pHCompressedTemplate	[out] Pointer to HNLTemplate that receives handle to created NLTTemplate object.

Returns

If the function succeeds, the return value is `N_OK` ([see page 82](#)).

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

Error code	Condition
<code>N_E_ARGUMENT_NULL</code> (see page 78)	<i>hExtractor</i> or <i>hTemplate</i> or <i>pHCompressedTemplate</i> is NULL (see page 133).

Notes

This function is obsolete. Use `NleCompressEx` ([see page 173](#)) instead.

Module

NLExtractor Module ([see page 171](#))

6.1.4.3.1.11 NleCreate Function

Creates a NLExtractor.

C++

```
NResult N_API NleCreate(HNLExtractor * pHExtractor);
```

Parameters

Parameters	Description
HNLExtractor * pHExtractor	[out] Pointer to HNLExtractor that receives handle to created NLExtractor object.

Returns

If the function succeeds, the return value is `N_OK` ([see page 82](#)).

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

Error code	Condition
<code>N_E_ARGUMENT_NULL</code> (see page 78)	<i>pHExtractor</i> is NULL (see page 133).
<code>N_E_OUT_OF_MEMORY</code> (see page 81)	There was not enough memory.

Remarks

Created object must be deleted using `NObjectFree` ([see page 97](#)) function.

Module

NLExtractor Module ([see page 171](#))

6.1.4.3.1.12 NleExtractStart Function

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

C++

```
NResult N_API NleExtractStart(HNLExtractor hExtractor, NInt durationInFrames);
```

Parameters

Parameters	Description
HNLExtractor hExtractor	[in] Handle to the NLExtractor object.
NInt durationInFrames	[in] Maximum number of frames which will be used for one feature template extraction.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>durationInFrames</i> is smaller than 1.

Module

NLExtractor Module ([see page 171](#))

6.1.4.3.1.13 NleGeneralize Function

Generates generalized template from uncompressed set of templates.

C++

```
NResult N_API NleGeneralize(HNLExtractor hExtractor, NInt templateCount, const HNLTTemplate * arHTemplates, HNLTTemplate * pHTemplate);
```

Parameters

Parameters	Description
HNLExtractor <i>hExtractor</i>	[in] Handle to the NLExtractor object.
NInt <i>templateCount</i>	[in] Count of templates to be used in generalization.
const HNLTTemplate * <i>arHTemplates</i>	[out] Pointer to memory buffer that contains NLTemplate templates.
HNLTTemplate * <i>pHTemplate</i>	[out] Pointer to HNLTTemplate that receives handle to created NLTemplate object.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hExtractor</i> , <i>arHTemplates</i> , <i>arTemplateSizes</i> or <i>pHTemplate</i> is NULL (see page 133).
N_E_ARGUMENT (see page 78)	One or more of templates in <i>arTemplates</i> might be NULL (see page 133) or compressed.

Notes

This function is obsolete. Use NleGeneralizeEx ([see page 178](#)) instead.

Module

NLExtractor Module ([see page 171](#))

6.1.4.3.1.14 HNLExtractor

Handle to NLExtractor object.

Remarks

The following parameters can be passed to function with corresponding handle:

- NLEP_MIN_IOD ([see page 186](#))

- NLEP_MAX_IOD (see page 185)
- NLEP_FACE_CONFIDENCE_THRESHOLD (see page 184)
- NLEP_FACE_QUALITY_THRESHOLD (see page 185)
- NLEP_GENERALIZATION_THRESHOLD

Module

NLExtractor Module (see page 171)

6.1.4.3.2 Structs, Records, Enums

6.1.4.3.2.1 NleRotation Structure

Enumerates face rotation values.

C++

```
typedef struct NleRotation_ {  
    NShort Yaw;  
    NShort Pitch;  
    NShort Roll;  
} NleRotation;
```

Members

Members	Description
NShort Yaw;	Rotation about vertical (y) axis.
NShort Pitch;	Rotation about the horizontal side-to-side (x) horizontal axis.
NShort Roll;	Rotation about the horizontal back-to-front (z) axis.

Module

NLExtractor Module (see page 171)

6.1.4.3.2.2 NleTemplateSize Enumeration

Enumerates available template size values.

C++

```
typedef enum NleTemplateSize_ {  
    nletsSmall = 0,  
    nletsMedium = 64,  
    nletsLarge = 128  
} NleTemplateSize;
```

Members

Members	Description
nletsSmall = 0	The small faces template size. This template should be used when the highest matching speed is required.
nletsMedium = 64	Medium faces template size. This is a default size for identification task.
nletsLarge = 128	Large template size. This template size should be used when the highest matching accuracy is needed.

Module

NLExtractor Module (see page 171)

6.1.4.3.2.3 NleDetectionDetails Structure

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

C++

```
typedef struct NleDetectionDetails_ {  
    NBool FaceAvailable;  
    NleFace Face;  
    NBool EyesAvailable;  
    NleEyes Eyes;  
} NleDetectionDetails;
```

Members

Members	Description
NBool FaceAvailable;	Variable showing if a face region was found in an image.
NleFace Face;	Structure with face region information if face was found.
NBool EyesAvailable;	Variable showing if eyes were found in the face region pointed by the face.
NleEyes Eyes;	Structure with found eyes information.

Module

NLExtractor Module (see page 171)

6.1.4.3.2.4 NleEyes Structure

Structure defining information of an eye pair.

C++

```
typedef struct NleEyes_ {  
    NPoint First;  
    NDouble FirstConfidence;  
    NPoint Second;  
    NDouble SecondConfidence;  
} NleEyes;
```

Members

Members	Description
NPoint First;	First eye coordinates.
NDouble FirstConfidence;	How confidently the first eye was found.
NPoint Second;	Second eye coordinates.
NDouble SecondConfidence;	How confidently the second eye was found.

Module

NLExtractor Module (see page 171)

6.1.4.3.2.5 NleExtractionStatus Enumeration

Face features extraction status.

C++

```
typedef enum NleExtractionStatus_ {  
    nleesNone = 0,  
    nleesTemplateCreated = 1,  
    nleesFaceNotDetected = 2,  
    nleesEyesNotDetected = 3,  
    nleesFaceTooCloseToImageBorder = 4,  
    nleesQualityCheckGrayscaleDensityFailed = 100,  
    nleesQualityCheckExposureFailed = 101,  
}
```

```

nleesQualityCheckSharpnessFailed = 102,
nleesLivenessCheckFailed = 200,
nleesGeneralizationFailed = 300
} NleExtractionStatus;

```

Members

Members	Description
nleesNone = 0	No extraction status usage.
nleesTemplateCreated = 1	Template created.
nleesFaceNotDetected = 2	Face not detected.
nleesEyesNotDetected = 3	Eyes not detected.
nleesFaceTooCloseToImageBorder = 4	Face too close to image border.
nleesQualityCheckGrayscaleDensityFailed = 100	Quality check grayscale density failed.
nleesQualityCheckExposureFailed = 101	Quality check exposure failed.
nleesQualityCheckSharpnessFailed = 102	Quality check sharpness failed.
nleesLivenessCheckFailed = 200	Liveness check failed.
nleesGeneralizationFailed = 300	Indicate that function failed to generalize extracted features.

Module

NLExtractor Module ([see page 171](#))

6.1.4.3.2 NleFace Structure

Structure describing face region information

C++

```

typedef struct NleFace_ {
    NRect Rectangle;
    NleRotation Rotation;
    NDouble Confidence;
} NleFace;

```

Members

Members	Description
NRect Rectangle;	Structure with face rectangle information.
NleRotation Rotation;	Face rotation structure.
NDouble Confidence;	How confidently the face region was found.

Module

NLExtractor Module ([see page 171](#))

6.1.4.3.3 Macros

To get a parameter value, use NObjectGetParameter ([see page 97](#)) function. To set parameter value - NObjectSetParameter ([see page 99](#)) function.

The following values can be used for *parameterId*:

- NLEP_MIN_IOD ([see page 186](#))
- NLEP_MAX_IOD ([see page 185](#))
- NLEP_FACE_CONFIDENCE_THRESHOLD ([see page 184](#))
- NLEP_FACE_QUALITY_THRESHOLD ([see page 185](#))
- NLEP_GENERALIZATION_THRESHOLD.

Macros

Name	Description
NLEP_FACE_CONFIDENCE_THRESHOLD (see page 184)	Identifier of type N_TYPE_DOUBLE (see page 103). Specifies the threshold which is considered when looking for faces in an image. For each face candidate confidence parameter is calculated. With higher threshold value faces are selected more strictly by the face detection routines. Must be in range [0..100].
NLEP_MAX_ROLL_ANGLE_DEVIATION (see page 184)	Defines maximum roll angle deviation from frontal face in degrees which is considered when looking for faces in an image. Must be in range [0..180].
NLEP_TEMPLATE_SIZE (see page 185)	Defines template size of Face template (NLTemplate).
NLEP_FACE_QUALITY_THRESHOLD (see page 185)	Identifier of type N_TYPE_BYTE (see page 102). Specifies the threshold which is considered when extracting facial features from the image. With higher threshold better quality of face image is required to successfully extract facial features. The value of this parameter can be in range [0..255]. The default value is 128.
NLEP_FAVOR_LARGEST_FACE (see page 185)	Identifier of type N_TYPE_BOOL (see page 102). If set to true the largest face found in an image will be used for facial feature extraction instead of the one with the highest face confidence threshold score.
NLEP_LIVENESS_THRESHOLD (see page 185)	Identifier of type N_TYPE_DOUBLE (see page 103). This threshold checks if faces are extracted from a live image stream. The higher the value of this parameter, the more strictly the stream is checked if the face in it is real (for example if it's not forged by showing a photo of a person in front a camera). The value of this parameter can be in range [0..100]. The default value is 50.
NLEP_MAX_IOD (see page 185)	Identifier of type N_TYPE_INT (see page 103) specifying maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].
NLEP_MAX_RECORDS_PER_TEMPLATE (see page 186)	Identifier of type N_TYPE_INT (see page 103). Sets the maximum number of records an extraction function can return in one NLTemplate. The value of this parameter can be in range [1..16]. The default value is 5.
NLEP_MIN_IOD (see page 186)	Identifier of type N_TYPE_INT (see page 103) specifying minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].
NLEP_USE_LIVENESS_CHECK (see page 186)	Identifier of type N_TYPE_BOOL (see page 102). If set to true liveness check is performed while doing facial feature extraction from an image stream.

Module

NLExtractor Module (see page 171)

6.1.4.3.3.1 NLEP_FACE_CONFIDENCE_THRESHOLD Macro

Identifier of type N_TYPE_DOUBLE (see page 103). Specifies the threshold which is considered when looking for faces in an image. For each face candidate confidence parameter is calculated. With higher threshold value faces are selected more strictly by the face detection routines. Must be in range [0..100].

C++

```
#define NLEP_FACE_CONFIDENCE_THRESHOLD 10103
```

6.1.4.3.3.2 NLEP_MAX_ROLL_ANGLE_DEVIATION Macro

Defines maximum roll angle deviation from frontal face in degrees which is considered when looking for faces in an image. Must be in range [0..180].

C++

```
#define NLEP_MAX_ROLL_ANGLE_DEVIATION 10105
```

Remarks

Matching speed highly depends on a roll angle deviation from frontal face.

6.1.4.3.3.3 NLEP_TEMPLATE_SIZE Macro

Defines template size of Face template (NLTemplate).

C++

```
#define NLEP_TEMPLATE_SIZE 10360
```

Remarks

Small, medium or large template can be used. Matching speed and accuracy depends on chosen template size. The maximum speed is achieved when small template is used. But the most accurate matching results are when large template is used.

By default when enrolling large template is saved to database. But when performing identification, by default medium template size is used.

Small, medium and large templates are fully compatible with older versions of VeriLook SDK.

See Also

For more information about template sizes see NleTemplateSize (see page 181) enumeration.

6.1.4.3.3.4 NLEP_FACE_QUALITY_THRESHOLD Macro

Identifier of type N_TYPE_BYTE (see page 102). Specifies the threshold which is considered when extracting facial features from the image. With higher threshold better quality of face image is required to successfully extract facial features. The value of this parameter can be in range [0..255]. The default value is 128.

C++

```
#define NLEP_FACE_QUALITY_THRESHOLD 10350
```

6.1.4.3.3.5 NLEP_FAVOR_LARGEST_FACE Macro

Identifier of type N_TYPE_BOOL (see page 102). If set to true the largest face found in an image will be used for facial feature extraction instead of the one with the highest face confidence threshold score.

C++

```
#define NLEP_FAVOR_LARGEST_FACE 10104
```

6.1.4.3.3.6 NLEP_LIVENESS_THRESHOLD Macro

Identifier of type N_TYPE_DOUBLE (see page 103). This threshold checks if faces are extracted from a live image stream. The higher the value of this parameter, the more strictly the stream is checked if the face in it is real (for example if it's not forged by showing a photo of a person in front a camera). The value of this parameter can be in range [0..100]. The default value is 50.

C++

```
#define NLEP_LIVENESS_THRESHOLD 10403
```

6.1.4.3.3.7 NLEP_MAX_IOD Macro

Identifier of type N_TYPE_INT (see page 103) specifying maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].

C++

```
#define NLEP_MAX_IOD 10102
```

6.1.4.3.3.8 NLEP_MAX_RECORDS_PER_TEMPLATE Macro

Identifier of type N_TYPE_INT (see page 103). Sets the maximum number of records an extraction function can return in one NLTemplate. The value of this parameter can be in range [1..16]. The default value is 5.

C++

```
#define NLEP_MAX_RECORDS_PER_TEMPLATE 10408
```

6.1.4.3.3.9 NLEP_MIN_IOD Macro

Identifier of type N_TYPE_INT (see page 103) specifying minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].

C++

```
#define NLEP_MIN_IOD 10101
```

6.1.4.3.3.10 NLEP_USE_LIVENESS_CHECK Macro

Identifier of type N_TYPE_BOOL (see page 102). If set to true liveness check is performed while doing facial feature extraction from an image stream.

C++

```
#define NLEP_USE_LIVENESS_CHECK 10402
```

6.1.4.3.4 Files

6.1.4.3.4.1 NLExtractor.h

Header file for the NLExtractor library. NLExtractor is a part of NLExtractor library intended for human face finding in pictures and extracting their features into Neurotechnology Faces Template(NLTemplate).

Enumerations

	Name	Description
	NleExtractionStatus_ (see page 182)	Face features extraction status.
	NleExtractionStatus (see page 182)	Face features extraction status.

Functions

	Name	Description
	NleCompress (see page 178)	Compresses the features extracted or generalized by the extraction routines to be stored or used in matching.
	NleCompressEx (see page 173)	Compresses the features extracted or generalized by the extraction routines to be stored or used in matching.
	NleCreate (see page 179)	Creates a NLExtractor.
	NleDetectFace (see page 174)	Finds one face region that best fits extraction requirements from the given image.
	NleDetectFaces (see page 174)	Finds all face regions in the given image.
	NleDetectFacialFeatures (see page 175)	Finds locations of facial feature points.
	NleExtract (see page 175)	Fully automatically extracts feature template from the face that best fits extraction requirements from the given image.

	NleExtractNext (see page 176)	Extracts one features template from image sequence
	NleExtractNextEx (see page 177)	Extracts one features template from image sequence.
	NleExtractStart (see page 179)	Initializes features extraction routines that use image sequences to extract features.
	NleExtractUsingDetails (see page 177)	Extracts feature template from the face with given detection details from an image.
	NleGeneralize (see page 180)	Generates generalized template from uncompressed set of templates.
	NleGeneralizeEx (see page 178)	Generates generalized template from uncompressed set of templates.

Module

NLExtractor Module ([see page 171](#))

Structures

	Name	Description
	NleDetectionDetails_ (see page 182)	Structure with information about face detection results in a face detection routine.
	NleEyes_ (see page 182)	Structure defining information of an eye pair.
	NleFace_ (see page 183)	Structure describing face region information
	NleRotation_ (see page 181)	Enumerates face rotation values.
	NleDetectionDetails (see page 182)	Structure with information about face detection results in a face detection routine.
	NleEyes (see page 182)	Structure defining information of an eye pair.
	NleFace (see page 183)	Structure describing face region information
	NleRotation (see page 181)	Enumerates face rotation values.

6.1.4.3.4.2 NLExtractorParams.h

C header file which contains NLExtractor parameters.

Enumerations

	Name	Description
	NleTemplateSize_ (see page 181)	Enumerates available template size values.
	NleTemplateSize (see page 181)	Enumerates available template size values.

Macros

Name	Description
NLEP_FACE_CONFIDENCE_THRESHOLD (see page 184)	Identifier of type N_TYPE_DOUBLE (see page 103). Specifies the threshold which is considered when looking for faces in an image. For each face candidate confidence parameter is calculated. With higher threshold value faces are selected more strictly by the face detection routines. Must be in range [0..100].
NLEP_FACE_QUALITY_THRESHOLD (see page 185)	Identifier of type N_TYPE_BYTE (see page 102). Specifies the threshold which is considered when extracting facial features from the image. With higher threshold better quality of face image is required to successfully extract facial features. The value of this parameter can be in range [0..255]. The default value is 128.
NLEP_FAVOR_LARGEST_FACE (see page 185)	Identifier of type N_TYPE_BOOL (see page 102). If set to true the largest face found in an image will be used for facial feature extraction instead of the one with the highest face confidence threshold score.

NLEP_LIVENESS_THRESHOLD (see page 185)	Identifier of type N_TYPE_DOUBLE (see page 103). This threshold checks if faces are extracted from a live image stream. The higher the value of this parameter, the more strictly the stream is checked if the face in it is real (for example if it's not forged by showing a photo of a person in front a camera). The value of this parameter can be in range [0..100]. The default value is 50.
NLEP_MAX_IOD (see page 185)	Identifier of type N_TYPE_INT (see page 103) specifying maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].
NLEP_MAX_RECORDS_PER_TEMPLATE (see page 186)	Identifier of type N_TYPE_INT (see page 103). Sets the maximum number of records an extraction function can return in one NLTemplate. The value of this parameter can be in range [1..16]. The default value is 5.
NLEP_MAX_ROLL_ANGLE_DEVIATION (see page 184)	Defines maximum roll angle deviation from frontal face in degrees which is considered when looking for faces in an image. Must be in range [0..180].
NLEP_MIN_IOD (see page 186)	Identifier of type N_TYPE_INT (see page 103) specifying minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].
NLEP_TEMPLATE_SIZE (see page 185)	Defines template size of Face template (NLTemplate).
NLEP_USE_LIVENESS_CHECK (see page 186)	Identifier of type N_TYPE_BOOL (see page 102). If set to true liveness check is performed while doing facial feature extraction from an image stream.

Module

NLExtractor Module (see page 171)

6.1.5 NImages Library

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

Remarks**Requirements (Windows)**

- **Import Library:** NImages.dll.lib
- **DLLs:**
 - NImages.dll
 - NCore.dll (see page 74)

Requirements (Linux)

- **Shared objects:**
 - libNImages.so
 - libNcore.so

Modules

Name	Description
Bmp Module (see page 189)	Provides functionality for loading and saving images in BMP format.
IHead Module (see page 194)	Provides functionality for loading and saving images in NIST IHead format.
Jpeg Module (see page 198)	Provides functionality for loading and saving images in JPEG format.
NGrayscaleImage Module (see page 204)	Provides functionality for managing 8-bit grayscale images.

NImage Module (see page 205)	Provides functionality for managing images.
NImageFile Module (see page 223)	Provides functionality for reading image files in format-neutral way.
NImages Module (see page 227)	Provides library registration and other additional functionality.
NMonochromeImage Module (see page 228)	Creates color wrapper for grayscale image.
NPixelFormat Module (see page 230)	Provides functionality for working with image pixel format.
NRgbImage Module (see page 235)	Provides functionality for managing 24-bit RGB images.
Png Module (see page 237)	Provides functionality for loading and saving images in PNG format.
Tiff Module (see page 242)	Provides functionality for loading images in TIFF format.
NImageFormat Module (see page 244)	Provides functionality for loading and saving images in format-neutral way.

6.1.5.1 Bmp Module

Provides functionality for loading and saving images in BMP format.

Files

Name	Description
Bmp.h (see page 194)	Header file for Bmp module. Provides functionality for loading and saving images in BMP format.

Functions

	Name	Description
≡	BmpLoadImageFromFile (see page 189)	Loads image from BMP file.
≡	BmpLoadImageFromHBitmap (see page 190)	Loads image from Windows HBITMAP.
≡	BmpLoadImageFromMemory (see page 190)	Loads image from memory buffer containing BMP file.
≡	BmpLoadImageFromStream (see page 191)	Loads image from stream containing Bmp file.
≡	BmpSaveImageToFile (see page 191)	Saves image to file in BMP format.
≡	BmpSaveImageToHBitmap (see page 192)	Saves image to Windows HBITMAP.
≡	BmpSaveImageToMemory (see page 192)	Saves image to memory buffer in BMP format.
≡	BmpSaveImageToStream (see page 193)	Saves image to stream in BMP format.

6.1.5.1.1 Functions

6.1.5.1.1.1 BmpLoadImageFromFile Function

Loads image from BMP file.

C++

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

Parameters

Parameters	Description
const NChar * szFileName	[in] Points to string that specifies file name.

HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when szFileName or pHImage is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. It is returned when format of file specified by szFileName is invalid.

Remarks

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

Module

Bmp Module (see page 189)

6.1.5.1.1.2 BmpLoadImageFromHBitmap Function

Loads image from Windows HBITMAP.

C++

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

Parameters

Parameters	Description
NHandle handle	[in] Handle that specifies Windows HBITMAP.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when handle or pHImage is NULL (see page 133).

Remarks

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

Notes

This function is available only on Windows.

Module

Bmp Module (see page 189)

6.1.5.1.1.3 BmpLoadImageFromMemory Function

Loads image from memory buffer containing BMP file.

C++

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

Parameters

Parameters	Description
const void * buffer	[out] Pointer to memory buffer.
NSizeType bufferSize	[in] Length of memory buffer.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when buffer is NULL (see page 133) and bufferSize is not equal to zero.- or -pHImage is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. It is returned when format of file contained in buffer specified by buffer is invalid.

Remarks

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

Module

Bmp Module (see page 189)

6.1.5.1.1.4 BmpLoadImageFromStream Function

Loads image from stream containing Bmp file.

C++

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

Parameters

Parameters	Description
HNStream hStream	[in] Handle to HNStream containing Bmp image.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).

Module

Bmp Module (see page 189)

6.1.5.1.1.5 BmpSaveImageToFile Function

Saves image to file in BMP format.

C++

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


Parameters

Parameters	Description
HNIImage hImage	[in] Handle to image.
const NChar * szFileName	[out] Points to string that specifies file name.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when hImage (see page 142) or szFileName is NULL (see page 133).

Remarks

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

Module

Bmp Module (see page 189)

6.1.5.1.1.6 BmpSaveImageToHBitmap Function

Saves image to Windows HBITMAP.

C++

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

Parameters

Parameters	Description
HNIImage hImage	[in] Handle to image.
NHandle * pHandle	[out] Pointer to NHandle (see page 117) that receives handle to created Windows HBITMAP.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when hImage (see page 142) or pHandle is NULL (see page 133).

Remarks

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

HBITMAP must be freed using GDI function DeleteObject.

Notes

This function is available only on Windows.

Module

Bmp Module (see page 189)

6.1.5.1.1.7 BmpSaveImageToMemory Function

Saves image to memory buffer in BMP format.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
void * * pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
NSizeType * pBufferLength	[out] Pointer to NSizeType (see page 119) that receives size of allocated memory buffer.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when hImage (see page 142), pBuffer or pBufferLength is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. It is returned when there was not enough memory to allocate memory buffer.

Remarks

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

Memory buffer allocated by the function must be deallocated using NFree (see page 94) function when it is no longer needed.

Module

Bmp Module (see page 189)

6.1.5.1.1.8 BmpSaveImageToStream Function

Saves image to stream in BMP format.

C++

```
NResult N_API BmpSaveImageToStream(HNImage hImage, HNStream hStream);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to HNStream containing Bmp image.
HNStream hStream	[out] Handle to HNStream where image will be written.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).

Module

Bmp Module (see page 189)

6.1.5.1.2 Files

6.1.5.1.2.1 Bmp.h

Header file for Bmp module. Provides functionality for loading and saving images in BMP format.

Functions

	Name	Description
⇒	BmpLoadImageFromFile (see page 189)	Loads image from BMP file.
⇒	BmpLoadImageFromHBitmap (see page 190)	Loads image from Windows HBITMAP.
⇒	BmpLoadImageFromMemory (see page 190)	Loads image from memory buffer containing BMP file.
⇒	BmpLoadImageFromStream (see page 191)	Loads image from stream containing Bmp file.
⇒	BmpSaveImageToFile (see page 191)	Saves image to file in BMP format.
⇒	BmpSaveImageToHBitmap (see page 192)	Saves image to Windows HBITMAP.
⇒	BmpSaveImageToMemory (see page 192)	Saves image to memory buffer in BMP format.
⇒	BmpSaveImageToStream (see page 193)	Saves image to stream in BMP format.

Module

Bmp Module (see page 189)

6.1.5.2 IHead Module

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

Files

Name	Description
IHead.h (see page 198)	Header file for IHead Module. Provides functionality for loading and saving images in NIST IHead format.

Functions

	Name	Description
⇒	IHeadLoadImageFromFile (see page 195)	Loads image from NIST IHead file.
⇒	IHeadLoadImageFromMemory (see page 195)	Loads image from memory buffer containing NIST IHead file.
⇒	IHeadLoadImageFromStream (see page 196)	Loads image data from memory stream.
⇒	IHeadSaveImageToFile (see page 196)	Saves image to the file in NIST IHead format.
⇒	IHeadSaveImageToMemory (see page 197)	Saves image to the memory buffer in NIST IHead format.
⇒	IHeadSaveImageToStream (see page 197)	Saves an image to specified memory stream.

6.1.5.2.1 Functions

6.1.5.2.1.1 IHeadLoadImageFromFile Function

Loads image from NIST IHead file.

C++

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

Parameters

Parameters	Description
const NChar * szFileName	[in] Points to string that specifies file name.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when szFileName or pHImage is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. It is returned when format of file specified by szFileName is invalid.

Remarks

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

Module

IHead Module (see page 194)

6.1.5.2.1.2 IHeadLoadImageFromMemory Function

Loads image from memory buffer containing NIST IHead file.

C++

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

Parameters

Parameters	Description
const void * buffer	[in] Pointer to memory buffer.
NSizeType bufferLength	[in] Length of memory buffer.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when: buffer is NULL (see page 133) and bufferLength is not equal to zero. - or - pHImage is NULL (see page 133).

N_E_FORMAT (see page 80)	If the function fails, this error can be returned. It is returned when format of file contained in buffer specified by buffer is invalid.

Remarks

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

Module

IHead Module (see page 194)

6.1.5.2.1.3 IHeadLoadImageFromStream Function

Loads image data from memory stream.

C++

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

Parameters

Parameters	Description
HNStream hStream	[in] Handle to memory stream that contains image data.
HNImage * pHImage	[out] Pointer to a loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_NOT_SUPPORTED (see page 81)	Image format is not supported.

Module

IHead Module (see page 194)

6.1.5.2.1.4 IHeadSaveImageToFile Function

Saves image to the file in NIST IHead format.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
const NChar * szFileName	[in] Points to string that specifies file name.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when hImage (see page 142) or szFileName is NULL (see page 133).

Remarks

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

Notes

This function is used if compiling for platform without ANSI versions of the functions support.

Module

IHead Module ([see page 194](#))

6.1.5.2.1.5 IHeadSaveImageToMemory Function

Saves image to the memory buffer in NIST IHead format.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
void * * pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
NSizeType * pBufferLength	[out] Pointer to NSizeType (see page 119) that receives size of allocated memory buffer.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. It is returned when hImage (see page 142), pBuffer or pBufferLength is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. It is returned when there was not enough memory to allocate memory buffer.

Remarks

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

Memory buffer allocated by the function must be deallocated using NFree ([see page 94](#)) function when it is no longer needed.

Module

IHead Module ([see page 194](#))

6.1.5.2.1.6 IHeadSaveImageToStream Function

Saves an image to specified memory stream.

C++

```
NResult N_API IHeadSaveImageToStream(HNImage hImage, HNStream hStream);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image to save to stream.
HNStream hStream	[out] Handle to HNStream where image will be written.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_NOT_SUPPORTED (see page 81)	Image format is not supported.

Module

IHead Module ([see page 194](#))

6.1.5.2.2 Files

6.1.5.2.2.1 IHead.h

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

Functions

	Name	Description
	IHeadLoadImageFromFile (see page 195)	Loads image from NIST IHead file.
	IHeadLoadImageFromMemory (see page 195)	Loads image from memory buffer containing NIST IHead file.
	IHeadLoadImageFromStream (see page 196)	Loads image data from memory stream.
	IHeadSaveImageToFile (see page 196)	Saves image to the file in NIST IHead format.
	IHeadSaveImageToMemory (see page 197)	Saves image to the memory buffer in NIST IHead format.
	IHeadSaveImageToStream (see page 197)	Saves an image to specified memory stream.

Module

IHead Module ([see page 194](#))

6.1.5.3 Jpeg Module

Provides functionality for loading and saving images in JPEG format.

Files

Name	Description
Jpeg.h (see page 204)	Header file for Jpeg module. Provides functionality for loading and saving images in JPEG format.

Functions

	Name	Description
	JpegLoadImageFromFile (see page 199)	Loads image from JPEG file.
	JpegLoadImageFromMemory (see page 199)	Loads image from the memory buffer containing JPEG file.
	JpegLoadImageFromStream (see page 200)	Loads image from the stream containing JPEG file.

	JpegSaveImageToFile (see page 201)	Saves image to file in JPEG format.
	JpegSaveImageToMemory (see page 201)	Saves image to the memory buffer in JPEG format.
	JpegSaveImageToStream (see page 202)	Saves image to the stream in JPEG format.
	LosslessJpegSaveImageToFile (see page 202)	Saves specified by HNImage (see page 221) handle image to file.
	LosslessJpegSaveImageToMemory (see page 203)	Saves specified lossless JPEG image to memory.
	LosslessJpegSaveImageToStream (see page 203)	Saves specified lossless JPEG image to memory stream.

Macros

Name	Description
JPEG_DEFAULT_QUALITY (see page 204)	Specifies default JPEG quality.

6.1.5.3.1 Functions

6.1.5.3.1.1 JpegLoadImageFromFile Function

Loads image from JPEG file.

C++

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

Parameters

Parameters	Description
const NChar * szFileName	[in] Points to string that specifies file name.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>szFileName</i> or <i>pHImage</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file specified by <i>szFileName</i> is invalid.

Module

Jpeg Module ([see page 198](#))

6.1.5.3.1.2 JpegLoadImageFromMemory Function

Loads image from the memory buffer containing JPEG file.

C++

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


Parameters

Parameters	Description
const void * buffer	[in] Pointer to memory buffer.
NSizeType bufferSize	[in] Length of memory buffer.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when buffer is NULL (see page 133) and bufferSize is not equal to zero.- or -pHImage is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file contained in buffer specified by buffer is invalid.

Remarks

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

Module

Jpeg Module (see page 198)

6.1.5.3.1.3 JpegLoadImageFromStream Function

Loads image from the stream containing JPEG file.

C++

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

Parameters

Parameters	Description
HNStream hStream	[in] Handle to NStream.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>buffer</i> is NULL (see page 133) and <i>bufferLength</i> is not equal to zero.
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file contained in memory buffer specified by buffer is invalid.

Remarks

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

Module

Jpeg Module (see page 198)

6.1.5.3.1.4 JpegSaveImageToFile Function

Saves image to file in JPEG format.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
NInt quality	[in] Specifies quality of JPEG compression.
const NChar * szFileName	[in] Points to string that specifies file name.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>szFileName</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>quality</i> is less than 1 or greater than 100.

Remarks

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

Module

Jpeg Module (see page 198)

6.1.5.3.1.5 JpegSaveImageToMemory Function

Saves image to the memory buffer in JPEG format.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
NInt quality	[in] Specifies quality of JPEG compression.
void * * pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
NSizeType * pBufferLength	[out] Pointer to NSizeType (see page 119) that receives size of allocated memory buffer.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).

N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142), <i>pBuffer</i> or <i>pBufferLength</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when quality is less than 1 or greater than 100.
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. This error code is returned when there was not enough memory to allocate memory buffer.

Remarks

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

Memory buffer allocated by this function must be deallocated by `NFree` (see page 94) function when it is no longer needed.

Module

Jpeg Module (see page 198)

6.1.5.3.1.6 JpegSaveImageToStream Function

Saves image to the stream in JPEG format.

C++

```
NResult N_API JpegSaveImageToStream(HNImage hImage, NInt quality, HNStream hStream);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
NInt quality	[in] Specifies quality of JPEG compression.
HNStream hStream	[out] Handle to NStream where image will be written.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142), <i>pBuffer</i> or <i>pBufferLength</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when quality is less than 1 or greater than 100.
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. This error code is returned when there was not enough memory to allocate memory buffer.

Remarks

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

Memory buffer allocated by this function must be deallocated by `NFree` (see page 94) function when it is no longer needed.

Module

Jpeg Module (see page 198)

6.1.5.3.1.7 LosslessJpegSaveImageToFile Function

Saves specified by HNImage (see page 221) handle image to file.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to NImage object which contains image to be saved to file.
const NChar * szFileName	[in] Zero terminated string which contains file name where Jpeg image will be saved.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

Jpeg Module (see page 198)

6.1.5.3.1.8 LosslessJpegSaveImageToMemory Function

Saves specified lossless JPEG image to memory.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to JPEG image to be saved to memory.
void * * pBuffer	[in] Pointer to memory buffer to save image.
NSizeType * pBufferLength	[in] Length of memory buffer where JPEG image will be saved.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

Jpeg Module (see page 198)

6.1.5.3.1.9 LosslessJpegSaveImageToStream Function

Saves specified lossless JPEG image to memory stream.

C++

```
NResult N_API LosslessJpegSaveImageToStream(HNImage hImage, HNStream hStream);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to JPEG image to be saved to memory.
HNStream hStream	[out] Handle to NStream which will be used to save lossless JPEG image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

Jpeg Module (see page 198)

6.1.5.3.2 Macros

6.1.5.3.2.1 JPEG_DEFAULT_QUALITY Macro

Specifies default JPEG quality.

C++

```
#define JPEG_DEFAULT_QUALITY 75
```

Module

Jpeg Module ([see page 198](#))

6.1.5.3.3 Files

6.1.5.3.3.1 Jpeg.h

Header file for Jpeg module. Provides functionality for loading and saving images in JPEG format.

Functions

	Name	Description
	JpegLoadImageFromFile (see page 199)	Loads image from JPEG file.
	JpegLoadImageFromMemory (see page 199)	Loads image from the memory buffer containing JPEG file.
	JpegLoadImageFromStream (see page 200)	Loads image from the stream containing JPEG file.
	JpegSaveImageToFile (see page 201)	Saves image to file in JPEG format.
	JpegSaveImageToMemory (see page 201)	Saves image to the memory buffer in JPEG format.
	JpegSaveImageToStream (see page 202)	Saves image to the stream in JPEG format.
	LosslessJpegSaveImageToFile (see page 202)	Saves specified by HNImage (see page 221) handle image to file.
	LosslessJpegSaveImageToMemory (see page 203)	Saves specified lossless JPEG image to memory.
	LosslessJpegSaveImageToStream (see page 203)	Saves specified lossless JPEG image to memory stream.

Macros

Name	Description
JPEG_DEFAULT_QUALITY (see page 204)	Specifies default JPEG quality.

Module

Jpeg Module ([see page 198](#))

6.1.5.4 NGrayscaleImage Module

Provides functionality for managing 8-bit grayscale images.

Files

Name	Description
NGrayscaleImage.h (see page 205)	Header file for NGrayscaleImage module. Provides functionality for managing 8-bit grayscale images.

6.1.5.4.1 Files

6.1.5.4.1.1 NGrayscaleImage.h

Header file for NGrayscaleImage module. Provides functionality for managing 8-bit grayscale images.

Module

NGrayscaleImage Module ([see page 204](#))

6.1.5.5 NImage Module

Provides functionality for managing images.

Files

Name	Description
NImage.h (see page 222)	Header file for NImage module. Provides functionality for managing images.

Functions

	Name	Description
	NImageClone (see page 206)	Creates a new image that is a copy of specified image.
	NImageCopy (see page 206)	Copies specified rectangular area of the NImage to other NImage object.
	NImageCreate (see page 207)	Creates an image with specified pixel format, size, stride and resolution.
	NImageCreateFromData (see page 208)	Creates an image with specified pixel format, size, stride and resolution and copies specified pixels to it.
	NImageCreateFromDataPart (see page 209)	Creates an image with specified pixel format, size, stride and resolution to specified source rectangular area.
	NImageCreateFromFile (see page 210)	Creates (loads) an image from file with specified format.
	NImageCreateFromImage (see page 211)	Creates an image from specified image with specified pixel format and stride.
	NImageCreateFromImageEx (see page 212)	Creates an image from specified image with specified pixel format, stride and resolution.
	NImageCreateWrapper (see page 213)	Creates an image wrapper for specified image pixels with specified pixel format, size, stride and resolution.
	NImageCrop (see page 214)	Crops an image.
	NImageFlipDiagonally (see page 214)	Flips image diagonally.
	NImageFlipHorizontally (see page 215)	Flips image horizontally.
	NImageFlipVertically (see page 215)	Flips image vertically.
	NImageGetHeight (see page 216)	Retrieves height of the image.
	NImageGetHorzResolution (see page 216)	Retrieves horizontal resolution of the image.
	NImageGetPixelFormat (see page 216)	Retrieves pixel format of the image.
	NImageGetPixels (see page 217)	Retrieves pointer to memory block containing pixels of the image.
	NImageGetSize (see page 217)	Retrieves size of memory block containing pixels of the image.
	NImageGetStride (see page 218)	Retrieves stride (size of one row) of the image.

	NImageGetVertResolution (see page 219)	Retrieves vertical resolution of the image.
	NImageGetWidth (see page 219)	Retrieves width of the image.
	NImageRotateFlip (see page 219)	Rotates image using rotation type.
	NImageSaveToFile (see page 220)	Saves the image to the file of specified format.
	HNImage (see page 221)	Handle to image.

Structs, Records, Enums

	Name	Description
	NImageRotateFlipType_ (see page 221)	Enumerates different image rotation types.
	NImageRotateFlipType (see page 221)	Enumerates different image rotation types.

6.1.5.5.1 Functions

6.1.5.5.1.1 NImageClone Function

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

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
HNImage * pHClonedImage	[out] Pointer to HNImage (see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pHClonedImage</i> is NULL (see page 133).

Remarks

Created image must be deleted using NObjectFree ([see page 97](#)) function.

Module

NImage Module ([see page 205](#))

6.1.5.5.1.2 NImageCopy Function

Copies specified rectangular area of the NImage to other NImage object.

C++

```
NResult N_API NImageCopy(HNImage hSrcImage, NUInt left, NUInt top, NUInt width, NUInt height, HNImage hDstImage, NUInt dstLeft, NUInt dstTop);
```

Parameters

Parameters	Description
HNIImage hSrcImage	[in] Handle to NImage object to copy.
NUInt left	[in] The x coordinate of source rectangle.
NUInt top	[in] The y coordinate of source rectangle.
NUInt width	[in] The width of source rectangle.
NUInt height	[in] The height of source rectangle.
HNIImage hDstImage	[in] Handle to a destination NImage object. Specified rectangular area will be copied to this object.
NUInt dstLeft	[in] The x coordinate in the destination NImage object. Copied NImage will be placed starting by this coordinate.
NUInt dstTop	[in] The y coordinate in the destination NImage object. Copied NImage will be placed starting by this coordinate.

Returns

If the function succeeds the return value is N_OK (see page 82). Otherwise, one of the following error codes are returned:

Return values	Description
N_E_ARGUMENT_NULL (see page 78)	<i>hSrcImage</i> or <i>hDstImage</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	Specified wrong coordinates or image width or height.

Module

NImage Module (see page 205)

6.1.5.5.1.3 NImageCreate Function

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

C++

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	[in] Specifies pixel format of the image.
NUInt width	[in] Specifies width of the image.
NUInt height	[in] Specifies height of the image.
NSizeType stride	[in] Specifies stride of the image. Can be zero.
NFloat horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
NFloat vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
HNIImage * pHImage	[out] Pointer to HNIImage (see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pixelFormat</i> has invalid value. - or - <i>stride</i> is not zero and is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pHImage</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is less than zero.
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. This error code is returned when there was not enough memory.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride (see page 218) function.

Created image must be deleted using NObjectFree (see page 97) function.

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

Module

NImage Module (see page 205)

6.1.5.5.1.4 NImageCreateFromData Function

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

C++

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	[in] Specifies pixel format of the image.
NUInt width	[in] Specifies width of the image.
NUInt height	[in] Specifies height of the image.
NSizeType stride	[in] Specifies stride of the image. Can be zero.
NFloat horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
NFloat vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
NSizeType srcStride	[in] Specifies stride of pixels to be copied to the image.
const void * srcPixels	[in] Points to memory block containing pixels that to be copied to the image.

HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pixelFormat</i> has invalid value. - or - <i>stride</i> is not zero and is less than minimal value for specified pixel format and width. - or - <i>srcStride</i> is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>srcPixels</i> or <i>pHImage</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is less than zero.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride (see page 218) function.

Format of memory block *srcPixels* points to must be the same as described in NImageGetPixels (see page 217) function, only stride is equal to *srcStride*.

Created image must be deleted using NObjectFree (see page 97) function.

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

Module

NImage Module (see page 205)

6.1.5.5.1.5 NImageCreateFromDataPart Function

Creates an image with specified pixel format, size, stride and resolution to specified source rectangular area.

C++

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	[in] Specifies pixel format of the image.
NUInt width	[in] Specifies width of the image.
NUInt height	[in] Specifies height of the image.
NSizeType stride	[in] Specifies stride of the image. Can be zero.
NFloat horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.

NFloat vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
NSizeType srcStride	[in] Specifies stride of the source image. Can be zero.
const void * srcPixels	[out] Pixels of source image.
NUInt left	[in] X coordinate of the source image.
NUInt top	[in] Y coordinate of the source image.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pixelFormat</i> has invalid value. - or - <i>stride</i> is not zero and is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pHImage</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is less than zero.
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. This error code is returned when there was not enough memory.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride (see page 218) function.

Created image must be deleted using NObjectFree (see page 97) function.

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

Module

NImage Module (see page 205)

6.1.5.5.1.6 NImageCreateFromFile Function

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

C++

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

Parameters

Parameters	Description
const NChar * szFileName	[out] Points to string that specifies file name.
HNImageFormat hImageFormat	[in] Handle to the image format of the file. Can be NULL (see page 133).

HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>szFileName</i> or <i>pHImage</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file specified by <i>szFileName</i> is invalid for specified image format.
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> is NULL (see page 133) and none of supported image formats is registered with file extension of <i>szFileName</i> . - or - <i>hImageFormat</i> is NULL (see page 133) and image format registered with file extension of <i>szFileName</i> does not support reading. - or - Image format specified by <i>hImageFormat</i> does not support reading.

Remarks

If *hImageFormat* is NULL (see page 133) image format is selected by file extension of *szFileName*.

Created image must be deleted using *NObjectFree* (see page 97) function.

Module

NImage Module (see page 205)

6.1.5.5.1.7 NImageCreateFromImage Function

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

C++

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	[in] Specifies pixel format of the image.
NSizeType stride	[in] Specifies stride of the image. Can be zero.
HNImage hSrcImage	[in] Handle to image used as source for the image.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).

N_E_ARGUMENT (see page 78)	( If the function fails, this error can be returned. This error code is returned when <i>pixelFormat</i> has invalid value. - or - <i>stride</i> is not zero and is less than minimal value for specified pixel format and source image width.
N_E_ARGUMENT_NULL (see page 78)	( If the function fails, this error can be returned. This error code is returned when <i>hSrcImage</i> or <i>pHImage</i> is NULL ( see page 133).

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride ( see page 218) function.

Created image must be deleted using NObjectFree ( see page 97) function.

Module

NImage Module ( see page 205)

6.1.5.5.1.8 NImageCreateFromImageEx Function

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

C++

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	[in] Specifies pixel format of the image.
NSizeType stride	[in] Specifies stride of the image. Can be zero.
NFloat horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
NFloat vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
HNIImage hSrcImage	[in] Handle to image used as source for the image.
HNIImage * pHImage	[out] Pointer to HNIImage ( see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK ( see page 82)	If the function succeeds and <i>pValue</i> is not NULL ( see page 133), the return value is N_OK ( see page 82).
N_E_ARGUMENT ( see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pixelFormat</i> has invalid value. - or - <i>stride</i> is not zero and is less than minimal value for specified pixel format and source image width.
N_E_ARGUMENT_NULL ( see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hSrcImage</i> or <i>pHImage</i> is NULL ( see page 133).
N_E_ARGUMENT_OUT_OF_RANGE ( see page 78)	If the function fails, this error can be returned. This error code is returned when <i>horzResolution</i> or <i>vertResolution</i> is less than zero.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride (see page 218) function.

Created image must be deleted using NObjectFree (see page 97) function.

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

Module

NImage Module (see page 205)

6.1.5.5.1.9 NImageCreateWrapper Function

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

C++

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	[in] Specifies pixel format of the image.
NUInt width	[in] Specifies width of the image.
NUInt height	[in] Specifies height of the image.
NSizeType stride	[in] Specifies stride of the image.
NFloat horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
NFloat vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
void * pixels	[out] Points to memory block containing pixels for the image.
NBool ownsPixels	[in] Specifies whether pixels will be automatically deleted with the image (if set to NTrue (see page 133)).
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pixelFormat</i> has invalid value. - or - <i>stride</i> is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pixels</i> or <i>pHImage</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is less than zero.

Remarks

For more information on image stride see `NImageGetStride` (see page 218) function.

Format of memory block pixels points to must be the same as described in `NImageGetPixels` (see page 217) function.

Created image must be deleted using `NObjectFree` (see page 97) function.

pixels must not be deleted during lifetime of the image. If *ownsPixels* is `NTrue` (see page 133) then pixels will be automatically deleted with the image.

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

Module

NImage Module (see page 205)

6.1.5.5.1.10 NImageCrop Function

Crops an image.

C++

```
NResult N_API NImageCrop(HNImage hImage, NUInt left, NUInt top, NUInt width, NUInt height,
HNImage * pHResultImage);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image that should be cropped.
NUInt left	[in] X-coordinate of the cropping rectangle.
NUInt top	[in] Y-coordinate of the cropping rectangle.
NUInt width	[in] Width of the cropping rectangle.
NUInt height	[in] Height of the cropping rectangle.
HNImage * pHResultImage	[out] Pointer to HNImage (see page 221) that receives handle of the cropped image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pHResultImage</i> is NULL (see page 133).

Module

NImage Module (see page 205)

6.1.5.5.1.11 NImageFlipDiagonally Function

Flips image diagonally.

C++

```
NResult N_API NImageFlipDiagonally(HNImage hImage);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image that should be flipped diagonally.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) is NULL (see page 133).

Module

NImage Module ([see page 205](#))

6.1.5.5.1.12 NImageFlipHorizontally Function

Flips image horizontally.

C++

```
NResult N_API NImageFlipHorizontally(HNImage hImage);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image that should be flipped horizontally.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) is NULL (see page 133).

Module

NImage Module ([see page 205](#))

6.1.5.5.1.13 NImageFlipVertically Function

Flips image vertically.

C++

```
NResult N_API NImageFlipVertically(HNImage hImage);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image that should be flipped vertically.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) is NULL (see page 133).

Module

NImage Module ([see page 205](#))

6.1.5.5.1.14 NImageGetHeight Function

Retrieves height of the image.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
NUInt * pValue	[out] Pointer to NUInt (see page 119) that receives height of the image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).

Module

NImage Module (see page 205)

6.1.5.5.1.15 NImageGetHorzResolution Function

Retrieves horizontal resolution of the image.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
NFloat * pValue	[out] Pointer to NFloat (see page 117) that receives horizontal resolution in pixels per inch of the image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).

Remarks

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

Module

NImage Module (see page 205)

6.1.5.5.1.16 NImageGetPixelFormat Function

Retrieves pixel format of the image.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
NPixelFormat * pValue	[out] Pointer to NPixelFormat (see page 232) that receives pixel format of the image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).

Module

NImage Module (see page 205)

6.1.5.5.1.17 NImageGetPixels Function

Retrieves pointer to memory block containing pixels of the image.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
void * * pValue	[out] Pointer to void * that receives pointer to memory block containing pixels of the image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).

Remarks

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

For more information see NImageGetPixelFormat (see page 216), NImageGetWidth (see page 219), NImageGetHeight (see page 216), NImageGetStride (see page 218), and NImageGetSize (see page 217) functions.

Module

NImage Module (see page 205)

6.1.5.5.1.18 NImageGetSize Function

Retrieves size of memory block containing pixels of the image.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
NSizeType * pValue	[out] Pointer to NSizeType (see page 119) that receives size of memory block containing pixels of the image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).

Remarks

Size of memory block containing image pixels is equal to image height multiplied by image stride. For more information see NImageGetHeight (see page 216) and NImageGetStride (see page 218) functions.

Module

NImage Module (see page 205)

6.1.5.5.1.19 NImageGetStride Function

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

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
NSizeType * pValue	[out] Pointer to NSizeType (see page 119) that receives stride of the image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).

Remarks

Stride (size of one row) of the image depends on image pixel format and width. It cannot be less than value obtained with NPixelFormatGetRowSize (see page 233) macro with arguments obtained with NImageGetPixelFormat (see page 216) and NImageGetWidth (see page 219) functions.

Module

NImage Module (see page 205)

6.1.5.5.1.20 NImageGetVertResolution Function

Retrieves vertical resolution of the image.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
NFloat * pValue	[out] Pointer to NFloat (see page 117) that receives vertical resolution in pixels per inch of the image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).

Remarks

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

Module

NImage Module (see page 205)

6.1.5.5.1.21 NImageGetWidth Function

Retrieves width of the image.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to the image.
NUInt * pValue	[out] Pointer to NUInt (see page 119) that receives width of the image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).

Module

NImage Module (see page 205)

6.1.5.5.1.22 NImageRotateFlip Function

Rotates image using rotation type.

C++

```
NResult N_API NImageRotateFlip(HNImage hImage, NImageRotateFlipType rotateFlipType, HNImage * pHResultImage);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image that should be rotated.
NImageRotateFlipType rotateFlipType	[in] One of the rotation type value.
HNImage * pHResultImage	[out] Pointer to rotated image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pHResultImage</i> is NULL (see page 133).
N_E_ARGUMENT (see page 78)	This error is returned when <i>rotateFlipType</i> has an invalid value.

Module

NImage Module (see page 205)

6.1.5.5.1.23 NImageSaveToFile Function

Saves the image to the file of specified format.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to NImage object.
const NChar * szFileName	[out] Points to string that specifies file name.
HNImageFormat hImageFormat	[in] Handle to the image format of the file. Can be NULL (see page 133).

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>szFileName</i> is NULL (see page 133).

N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> is NULL (see page 133) and none of supported image formats is registered with file extension of <i>szFileName</i>. - or - <i>hImageFormat</i> is NULL (see page 133) and image format registered with file extension of <i>szFileName</i> does not support writing. - or - Image format specified by <i>hImageFormat</i> does not support writing.

Remarks

If *hImageFormat* is NULL ([see page 133](#)) image format is selected by file extension of *szFileName*.

Module

NImage Module ([see page 205](#))

6.1.5.5.1.24 HNImage

Handle to image.

Module

NImage Module ([see page 205](#))

6.1.5.5.2 Structs, Records, Enums**6.1.5.5.2.1 NImageRotateFlipType Enumeration**

Enumerates different image rotation types.

C++

```
typedef enum NImageRotateFlipType_ {
    nirftRotateNone = 0,
    nirftRotate90 = 1,
    nirftRotate180 = 2,
    nirftRotate270 = 3,
    nirftFlipNone = 0,
    nirftFlipX = 4,
    nirftFlipY = 8,
    nirftFlipXY = nirftFlipX|nirftFlipY,
    nirftNone = nirftRotateNone|nirftFlipNone
} NImageRotateFlipType;
```

Members

Members	Description
nirftRotateNone = 0	Indicates that image should not be rotated.
nirftRotate90 = 1	Rotates image by 90 degrees.
nirftRotate180 = 2	Rotates image by 180 degrees.
nirftRotate270 = 3	Rotates image by 270 degrees.
nirftFlipNone = 0	Indicates that image should not be flipped.
nirftFlipX = 4	Flips the image's orientation of the X coordinates.
nirftFlipY = 8	Flips the image's orientation of the Y coordinates.
nirftFlipXY = nirftFlipX nirftFlipY	Flips the image's orientation of the X or Y coordinates.
nirftNone = nirftRotateNone nirftFlipNone	Indicates that image should not be rotated or flipped.

Module

NImage Module ([see page 205](#))

6.1.5.5.3 Files**6.1.5.5.3.1 NImage.h**

Header file for NImage module. Provides functionality for managing images.

Enumerations

	Name	Description
	NImageRotateFlipType_ (see page 221)	Enumerates different image rotation types.
	NImageRotateFlipType (see page 221)	Enumerates different image rotation types.

Functions

	Name	Description
	NImageClone (see page 206)	Creates a new image that is a copy of specified image.
	NImageCopy (see page 206)	Copies specified rectangular area of the NImage to other NImage object.
	NImageCreate (see page 207)	Creates an image with specified pixel format, size, stride and resolution.
	NImageCreateFromData (see page 208)	Creates an image with specified pixel format, size, stride and resolution and copies specified pixels to it.
	NImageCreateFromDataPart (see page 209)	Creates an image with specified pixel format, size, stride and resolution to specified source rectangular area.
	NImageCreateFromFile (see page 210)	Creates (loads) an image from file with specified format.
	NImageCreateFromImage (see page 211)	Creates an image from specified image with specified pixel format and stride.
	NImageCreateFromImageEx (see page 212)	Creates an image from specified image with specified pixel format, stride and resolution.
	NImageCreateWrapper (see page 213)	Creates an image wrapper for specified image pixels with specified pixel format, size, stride and resolution.
	NImageCrop (see page 214)	Crops an image.
	NImageFlipDiagonally (see page 214)	Flips image diagonally.
	NImageFlipHorizontally (see page 215)	Flips image horizontally.
	NImageFlipVertically (see page 215)	Flips image vertically.
	NImageGetHeight (see page 216)	Retrieves height of the image.
	NImageGetHorzResolution (see page 216)	Retrieves horizontal resolution of the image.
	NImageGetPixelFormat (see page 216)	Retrieves pixel format of the image.
	NImageGetPixels (see page 217)	Retrieves pointer to memory block containing pixels of the image.
	NImageGetSize (see page 217)	Retrieves size of memory block containing pixels of the image.
	NImageGetStride (see page 218)	Retrieves stride (size of one row) of the image.
	NImageGetVertResolution (see page 219)	Retrieves vertical resolution of the image.
	NImageGetWidth (see page 219)	Retrieves width of the image.

	NImageRotateFlip (see page 219)	Rotates image using rotation type.
	NImageSaveToFile (see page 220)	Saves the image to the file of specified format.

Module

NImage Module ([see page 205](#))

6.1.5.6 NImageFile Module

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

Files

Name	Description
NImageFile.h (see page 226)	Header file for NImageFile module. Provides functionality for reading image files in format-neutral way.

Functions

	Name	Description
	NImageFileClose (see page 223)	Closes the file associated with the image file.
	NImageFileCreate (see page 224)	Opens image file of specified format.
	NImageFileGetFormat (see page 224)	Retrieves image format of the image file.
	NImageFileIsOpened (see page 225)	Retrieves a value indicating whether the file associated with the image file is opened.
	NImageFileReadImage (see page 225)	Reads image from the image file.
	HNImageFile (see page 226)	Handle to opened read-only image file.

6.1.5.6.1 Functions

6.1.5.6.1.1 NImageFileClose Function

Closes the file associated with the image file.

C++

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

Parameters

Parameters	Description
HNImageFile hImageFile	[in] Handle to image file.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFile</i> is NULL (see page 133).

Remarks

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

Module

NImageFile Module ([see page 223](#))

6.1.5.6.1.2 NImageFileCreate Function

Opens image file of specified format.

C++

```
NResult N_API NImageFileCreate(const NChar * szFileName, HImageFormat hImageFormat,
HImageFile * pHImageFile);
```

Parameters

Parameters	Description
const NChar * szFileName	[in] Points to string that specifies file name.
HImageFormat hImageFormat	[in] Handle to the image format of the file. Can be NULL (see page 133).
HImageFile * pHImageFile	[out] Pointer to HImageFile (see page 226) that receives handle to opened image file.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>szFileName</i> or <i>pHImageFile</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file specified by <i>szFileName</i> is invalid for specified image format.
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> is NULL (see page 133) and none of supported image formats is registered with file extension of <i>szFileName</i> . - or - <i>hImageFormat</i> is NULL (see page 133) and image format registered with file extension of <i>szFileName</i> does not support reading. - or - Image format specified by <i>hImageFormat</i> does not support reading.

Remarks

If *hImageFormat* is NULL ([see page 133](#)) image format is selected by file extension of *szFileName*.

Opened image file must be closed using NObjectFree ([see page 97](#)) function.

This function does not check format of the file for all image formats. However format of the file is always checked in NImageFileReadImage ([see page 225](#)) function.

Module

NImageFile Module ([see page 223](#))

6.1.5.6.1.3 NImageFileGetFormat Function

Retrieves image format of the image file.

C++

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

Parameters

Parameters	Description
HNImageFile hImageFile	[in] Handle to image file.
HNImageFormat * pValue	[out] Pointer to NImageFormat that receives image format of the image file.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFile</i> or <i>pValue</i> is NULL (see page 133).

Module

NImageFile Module (see page 223)

6.1.5.6.1.4 NImageFileIsOpened Function

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

C++

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

Parameters

Parameters	Description
HNImageFile hImageFile	[in] Handle to image file.
NBool * pValue	[out] Pointer to NBool (see page 116) that receives value indicating whether the file associated with the image file is opened.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFile</i> or <i>pValue</i> is NULL (see page 133).

Module

NImageFile Module (see page 223)

6.1.5.6.1.5 NImageFileReadImage Function

Reads image from the image file.

C++

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

Parameters

Parameters	Description
HNImageFile hImageFile	[in] Handle to image file.

HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to image read from the image file.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFile</i> or <i>pHImage</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file associated with the image file is invalid for the image format of the image file.
N_E_INVALID_OPERATION (see page 80)	If the function fails, this error can be returned. This error code is returned when file associated with the image file is closed.

Remarks

If all images are already read from the image file then handle to image returned via *pHImage* is NULL (see page 133).

Module

NImageFile Module (see page 223)

6.1.5.6.1.6 HNImageFile

Handle to opened read-only image file.

Module

NImageFile Module (see page 223)

6.1.5.6.2 Files**6.1.5.6.2.1 NImageFile.h**

Header file for NImageFile module. Provides functionality for reading image files in format-neutral way.

Functions

	Name	Description
≡	NImageFileClose (see page 223)	Closes the file associated with the image file.
≡	NImageFileCreate (see page 224)	Opens image file of specified format.
≡	NImageFileGetFormat (see page 224)	Retrieves image format of the image file.
≡	NImageFileIsOpened (see page 225)	Retrieves a value indicating whether the file associated with the image file is opened.
≡	NImageFileReadImage (see page 225)	Reads image from the image file.

Module

NImageFile Module (see page 223)

6.1.5.7 NImages Module

Provides library registration and other additional functionality.

Files

Name	Description
NImages.h (see page 228)	Header file for NImages (see page 188) module. Provides library registration and other additional functionality.

Functions

	Name	Description
◆	NImagesGetGrayscaleColorWrapperEx (see page 227)	Creates color wrapper for a grayscale image.
◆	NImagesGetInfo (see page 228)	Gets information about the library.

6.1.5.7.1 Functions

6.1.5.7.1.1 NImagesGetGrayscaleColorWrapperEx Function

Creates color wrapper for a grayscale image.

C++

```
NResult N_API NImagesGetGrayscaleColorWrapperEx(HNImage hImage, const NRGB * pMinColor, const NRGB * pMaxColor, HNImage * pHDstImage);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
const NRGB * pMinColor	[out] Specifies color to be used for black color.
const NRGB * pMaxColor	[out] Specifies color to be used for white color.
HNImage * pHDstImage	[out] Pointer to HNImage (see page 221) that receives handle to created image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT (see page 78)	This error code is returned when image specified by <i>hImage</i> (see page 142) has non-grayscale pixel format (not <i>npfGrayscale</i> or <i>npfMonochrome</i>).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pHDstImage</i> is NULL (see page 133).

Remarks

Created image must be deleted using NObjectFree (see page 97) function.

Created image is a thin wrapper for specified grayscale image. Therefore *hImage* (see page 142) must not be freed before created image.

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

Module

NImages Module (see page 227)

6.1.5.7.1.2 NImagesGetInfo Function

Gets information about the library.

C++

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

Parameters

Parameters	Description
NLibraryInfo * pValue	[out] Pointer to NLibraryInfo structure that receives library information.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pValue is NULL (see page 133).

Module

NImages Module (see page 227)

6.1.5.7.2 Files

6.1.5.7.2.1 NImages.h

Header file for NImages (see page 188) module. Provides library registration and other additional functionality.

Functions

	Name	Description
	NImagesGetGrayscaleColorWrapperEx (see page 227)	Creates color wrapper for a grayscale image.
	NImagesGetInfo (see page 228)	Gets information about the library.

Module

NImages Module (see page 227)

6.1.5.8 NMonochromeImage Module

Creates color wrapper for grayscale image.

Files

Name	Description
NMonochromeImage.h (see page 230)	Header file for NMonochromeImage module. Provides functionality for managing 1-bit monochrome images.

Functions

	Name	Description
	NMonochromeImageGetPixel ([2] see page 229)	Retrieves value of pixel at the specified coordinates in 1-bit monochrome image.
	NMonochromeImageSetPixel ([2] see page 229)	Sets value of pixel at the specified coordinates in 1-bit monochrome image.

6.1.5.8.1 Functions**6.1.5.8.1.1 NMonochromeImageGetPixel Function**

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

C++

```
NResult N_API NMonochromeImageGetPixel(HMonochromeImage hImage, NUInt x, NUInt y, NBool * pValue);
```

Parameters

Parameters	Description
HMonochromeImage hImage	[in] Handle to image.
NUInt x	[in] Specifies x-coordinate of the pixel.
NUInt y	[in] Specifies y-coordinate of the pixel.
NBool * pValue	[out] Points to NBool ([2] see page 116) that receives pixel value.

Returns

Return values	Description
N_OK ([2] see page 82)	If the function succeeds, the return value is N_OK ([2] see page 82).
N_E_ARGUMENT_NULL ([2] see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> ([2] see page 142) or <i>pValue</i> is NULL ([2] see page 133).
N_E_ARGUMENT_OUT_OF_RANGE ([2] see page 78)	This error code is returned when x is greater than or equal to image width. - or - y is greater than or equal to image height.
N_E_FORMAT ([2] see page 80)	This error code is returned when image pixel format is not equal to <i>npfMonochrome</i> .

Remarks

If pixel has black color then a value that *pValue* points to receives NFalse ([\[2\]](#) see page 133) and if it is white then value receives NTrue ([\[2\]](#) see page 133).

Module

NMonochromeImage Module ([\[2\]](#) see page 228)

6.1.5.8.1.2 NMonochromeImageSetPixel Function

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

C++

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

Parameters

Parameters	Description
HNMonochromeImage hImage	[in] Handle to image.
NUInt x	[in] Specifies x-coordinate of the pixel.
NUInt y	[in] Specifies y-coordinate of the pixel.
NBool value	[in] Specifies new pixel value.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	This error code is returned when <i>hImage</i> (see page 142) is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	This error code is returned when x is greater than or equal to image width. - or - y is greater than or equal to image height.
N_E_FORMAT (see page 80)	This error code is returned when image pixel format is not equal to <i>npfMonochrome</i> .

Remarks

If value is NFalse (see page 133) then pixel will be black and if it is NTrue (see page 133) then pixel will be white.

Module

NMonochromeImage Module (see page 228)

6.1.5.8.2 Files

6.1.5.8.2.1 NMonochromeImage.h

Header file for NMonochromeImage module. Provides functionality for managing 1-bit monochrome images.

Functions

	Name	Description
	NMonochromeImageGetPixel (see page 229)	Retrieves value of pixel at the specified coordinates in 1-bit monochrome image.
	NMonochromeImageSetPixel (see page 229)	Sets value of pixel at the specified coordinates in 1-bit monochrome image.

Module

NMonochromeImage Module (see page 228)

6.1.5.9 NPixelFormat Module

Provides functionality for working with image pixel format.

Files

Name	Description
NPixelFormat.h (see page 234)	Header file for NPixelFormat (see page 232) module. Provides functionality for working with image pixel format.

Functions

	Name	Description
	NPixelformatIsValid (see page 231)	Checks if specified pixel format is valid.

Macros

Name	Description
NCalcRowSize (see page 233)	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
NCalcRowSizeEx (see page 233)	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel and alignment.
NPixelformatGetBitsPerPixel (see page 233)	Retrieves number of bits used to store a pixel from NPixelFormat (see page 232).
NPixelformatGetRowSize (see page 233)	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat (see page 232).
NPixelformatGetRowSizeEx (see page 233)	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat (see page 232) and alignment.
NRgbConst (see page 234)	Makes NRGB (see page 232) constant with field values provided.

Structs, Records, Enums

	Name	Description
	NPixelformat_ (see page 232)	Specifies pixel format of each pixel in the image.
	NRGB_ (see page 232)	Specifies pixel format of each pixel in the image.
	NRGB (see page 232)	Specifies pixel format of each pixel in the image.

Types

Name	Description
NPixelformat (see page 232)	Specifies pixel format of each pixel in the image.

6.1.5.9.1 Functions

6.1.5.9.1.1 NPixelformatIsValid Function

Checks if specified pixel format is valid.

C++

```
NBool N_API NPixelformatIsValid(NPixelformat value);
```

Parameters

Parameters	Description
NPixelformat value	The NPixelFormat (see page 232) object.

Returns

True, if pixel format is valid; false otherwise.

Module

NPixelformat Module (see page 230)

6.1.5.9.2 Structs, Records, Enums

6.1.5.9.2.1 NPixelFormat_ Enumeration

Specifies pixel format of each pixel in the image.

C++

```
enum NPixelFormat_ {  
    npfMonochrome = 0x00001001,  
    npfGrayscale = 0x00301001,  
    npfRgb = 0x00303003  
};
```

Members

Members	Description
npfMonochrome = 0x00001001	Each pixel value is stored in 1 bit representing either black or white color.
npfGrayscale = 0x00301001	Each pixel value is stored in 8 bits representing 256 shades of gray.
npfRgb = 0x00303003	Each pixel value is stored in 24 bits consisting of three 8-bit values representing red, green and blue color components.

Remarks

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

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.2.2 NRgb Structure

Specifies pixel format of each pixel in the image.

C++

```
typedef struct NRgb_ {  
    NByte Red;  
    NByte Green;  
    NByte Blue;  
} NRgb;
```

Members

Members	Description
NByte Red;	Red component value of this NRgb_.
NByte Green;	Green component value of this NRgb_.
NByte Blue;	Blue component value of this NRgb_.

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.3 Types

6.1.5.9.3.1 NPixelFormat Type

Specifies pixel format of each pixel in the image.

C++

```
typedef enum NPixelFormat_ NPixelFormat;
```

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.4 Macros**6.1.5.9.4.1 NCalcRowSize Macro**

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

C++

```
#define NCalcRowSize(bitCount, length) NCalcRowSizeEx(bitCount, length, 1)
```

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.4.2 NCalcRowSizeEx Macro

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

C++

```
#define NCalcRowSizeEx(bitCount, length, alignment) (((bitCount) * (length) + ((alignment) < 3) - 1) / ((alignment) < 3)) * (alignment)
```

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.4.3 NPixelFormatGetBitsPerPixel Macro

Retrieves number of bits used to store a pixel from NPixelFormat ([see page 232](#)).

C++

```
#define NPixelFormatGetBitsPerPixel(pixelFormat) NPixelFormatGetBitsPerPixelFunc(pixelFormat)
```

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.4.4 NPixelFormatGetRowSize Macro

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat ([see page 232](#)).

C++

```
#define NPixelFormatGetRowSize(pixelFormat, length) NPixelFormatGetRowSizeEx(pixelFormat, length, 1)
```

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.4.5 NPixelFormatGetRowSizeEx Macro

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat ([see page 232](#)) and alignment.

C++

```
#define NPixelFormatGetRowSizeEx(pixelFormat, length, alignment)
NCalcRowSizeEx(NPixelFormatGetBitsPerPixel(pixelFormat), length, alignment)
```

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.4.6 NRgbConst Macro

Makes NRgb ([see page 232](#)) constant with field values provided.

C++

```
#define NRgbConst(red, green, blue) { red, green, blue }
```

Module

NPixelFormat Module ([see page 230](#))

6.1.5.9.5 Files**6.1.5.9.5.1 NPixelFormat.h**

Header file for NPixelFormat ([see page 232](#)) module. Provides functionality for working with image pixel format.

Enumerations

	Name	Description
	NPixelFormat_ (see page 232)	Specifies pixel format of each pixel in the image.

Functions

	Name	Description
	NPixelFormatIsValid (see page 231)	Checks if specified pixel format is valid.

Macros

Name	Description
NCalcRowSize (see page 233)	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
NCalcRowSizeEx (see page 233)	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel and alignment.
NPixelFormatGetBitsPerPixel (see page 233)	Retrieves number of bits used to store a pixel from NPixelFormat (see page 232).
NPixelFormatGetRowSize (see page 233)	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat (see page 232).
NPixelFormatGetRowSizeEx (see page 233)	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat (see page 232) and alignment.
NRgbConst (see page 234)	Makes NRgb (see page 232) constant with field values provided.

Module

NPixelFormat Module ([see page 230](#))

Structures

	Name	Description
	NRgb_ (see page 232)	Specifies pixel format of each pixel in the image.
	NRgb (see page 232)	Specifies pixel format of each pixel in the image.

Types

Name	Description
NPixelFormat (see page 232)	Specifies pixel format of each pixel in the image.

6.1.5.10 NRgbImage Module

Provides functionality for managing 24-bit RGB images.

Files

Name	Description
NRgbImage.h (see page 236)	Header file for NRgbImage module. Provides functionality for managing 24-bit RGB images.

Functions

	Name	Description
⇒	NRgbImageGetPixel (see page 235)	Retrieves value of pixel at the specified coordinates in 24-bit RGB image.
⇒	NRgbImageSetPixel (see page 236)	Sets value of pixel at the specified coordinates in 24-bit RGB image.

6.1.5.10.1 Functions**6.1.5.10.1.1 NRgbImageGetPixel Function**

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

C++

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

Parameters

Parameters	Description
HNRgbImage hImage	[in] Handle to image.
NUInt x	[in] Specifies x-coordinate of the pixel.
NUInt y	[in] Specifies y-coordinate of the pixel.
NRgb * pValue	[out] Pointer to NRgb (see page 232) that receives pixel value.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>x</i> is greater than or equal to image width. - or - <i>y</i> is greater than or equal to image height.
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when image pixel format is not equal to <i>npfRgb</i> .

Module

NRgbImage Module ([see page 235](#))

6.1.5.10.1.2 NRgbImageSetPixel Function

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

C++

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

Parameters

Parameters	Description
HNRgbImage hImage	[in] Handle to image.
NUInt x	[in] Specifies x-coordinate of the pixel.
NUInt y	[in] Specifies y-coordinate of the pixel.
const NRGB * pValue	[out] Pointer to NRGB (see page 232) that specifies new pixel value.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>x</i> is greater than or equal to image width. - or - <i>y</i> is greater than or equal to image height.
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when image pixel format is not equal to <i>npfRgb</i> .

Module

NRgbImage Module ([see page 235](#))

6.1.5.10.2 Files**6.1.5.10.2.1 NRgbImage.h**

Header file for NRgbImage module. Provides functionality for managing 24-bit RGB images.

Functions

	Name	Description
	NRgbImageGetPixel (see page 235)	Retrieves value of pixel at the specified coordinates in 24-bit RGB image.
	NRgbImageSetPixel (see page 236)	Sets value of pixel at the specified coordinates in 24-bit RGB image.

Module

NRgbImage Module ([see page 235](#))

6.1.5.11 Png Module

Provides functionality for loading and saving images in PNG format.

Files

Name	Description
Png.h (see page 241)	Header file for Png module. Provides functionality for loading and saving images in PNG format.

Functions

	Name	Description
⇒	PngLoadImageFromFile (see page 237)	Loads image from PNG file.
⇒	PngLoadImageFromMemory (see page 238)	Loads image from the memory buffer containing PNG file.
⇒	PngLoadImageFromStream (see page 238)	Loads image from the stream containing PNG file.
⇒	PngSaveImageToFile (see page 239)	Saves image to file in PNG format.
⇒	PngSaveImageToMemory (see page 239)	Saves image to the memory buffer in PNG format.
⇒	PngSaveImageToStream (see page 240)	Saves image to the stream in PNG format.

Macros

Name	Description
PNG_DEFAULT_COMPRESSION_LEVEL (see page 241)	Specifies default PNG compression level.

6.1.5.11.1 Functions

6.1.5.11.1.1 PngLoadImageFromFile Function

Loads image from PNG file.

C++

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

Parameters

Parameters	Description
const NChar * szFileName	[out] Points to string that specifies file name.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).

Remarks

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

Module

Png Module (see page 237)

6.1.5.11.1.2 PngLoadImageFromMemory Function

Loads image from the memory buffer containing PNG file.

C++

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

Parameters

Parameters	Description
const void * buffer	[in] Pointer to memory buffer.
NSizeType bufferLength	[in] Length of memory buffer.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>buffer</i> is NULL (see page 133) and <i>bufferLength</i> is not equal to zero or <i>pHImage</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file contained in memory buffer specified by <i>buffer</i> is invalid.

Remarks

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

Module

Png Module (see page 237)

6.1.5.11.1.3 PngLoadImageFromStream Function

Loads image from the stream containing PNG file.

C++

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

Parameters

Parameters	Description
HNStream hStream	[in] Handle to HNStream.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>buffer</i> is NULL (see page 133) and <i>bufferLength</i> is not equal to zero or <i>pHImage</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file contained in memory buffer specified by <i>buffer</i> is invalid.

Remarks

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

Module

Png Module (see page 237)

6.1.5.11.1.4 PngSaveImageToFile Function

Saves image to file in PNG format.

C++

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

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
NInt compressionLevel	[in] Specifies level of PNG compression. The value can be in range [0, 9]. 0 - uncompressed, 9 - maximal compression level. See PNG_DEFAULT_COMPRESSION_LEVEL (see page 241).
const NChar * szFileName	[out] Points to string that specifies file name.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142) or <i>szFileName</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>compressionLevel</i> is less than 0 or greater than 9.

Remarks

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

Module

Png Module (see page 237)

6.1.5.11.1.5 PngSaveImageToMemory Function

Saves image to the memory buffer in PNG format.

C++

```
NResult N_API PngSaveImageToMemory(HNImage hImage, NInt compressionLevel, void * * pBuffer,
NSizeType * pBufferLength);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
NInt compressionLevel	[in] Specifies level of PNG compression. The value can be in range [0, 9]. 0 - uncompressed, 9 - maximal compression level. See PNG_DEFAULT_COMPRESSION_LEVEL (see page 241).
void * * pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
NSizeType * pBufferLength	[out] Pointer to NSizeType (see page 119) that receives size of allocated memory buffer.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142), <i>pBuffer</i> or <i>pBufferLength</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>compressionLevel</i> is less than 0 or greater than 9.
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. This error code is returned when there was not enough memory to allocate memory buffer.

Remarks

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

Memory buffer allocated by this function must be deallocated by NFree (see page 94) function when it is no longer needed.

Module

Png Module (see page 237)

6.1.5.11.1.6 PngSaveImageToStream Function

Saves image to the stream in PNG format.

C++

```
NResult N_API PngSaveImageToStream(HNImage hImage, NInt compressionLevel, HNStream hStream);
```

Parameters

Parameters	Description
HNImage hImage	[in] Handle to image.
NInt compressionLevel	[in] Specifies level of PNG compression. The value can be in range [0, 9]. 0 - uncompressed, 9 - maximal compression level. See PNG_DEFAULT_COMPRESSION_LEVEL (see page 241).
HNStream hStream	[out] Handle to HNStream where image will be written.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImage</i> (see page 142), <i>pBuffer</i> or <i>pBufferLength</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>compressionLevel</i> is less than 0 or greater than 9.
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. This error code is returned when there was not enough memory to allocate memory buffer.

Remarks

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

Memory buffer allocated by this function must be deallocated by NFree (see page 94) function when it is no longer needed.

Module

Png Module (see page 237)

6.1.5.11.2 Macros

6.1.5.11.2.1 PNG_DEFAULT_COMPRESSION_LEVEL Macro

Specifies default PNG compression level.

C++

```
#define PNG_DEFAULT_COMPRESSION_LEVEL 6
```

Module

Png Module (see page 237)

6.1.5.11.3 Files

6.1.5.11.3.1 Png.h

Header file for Png module. Provides functionality for loading and saving images in PNG format.

Functions

	Name	Description
≡	PngLoadImageFromFile (see page 237)	Loads image from PNG file.
≡	PngLoadImageFromMemory (see page 238)	Loads image from the memory buffer containing PNG file.
≡	PngLoadImageFromStream (see page 238)	Loads image from the stream containing PNG file.
≡	PngSaveImageToFile (see page 239)	Saves image to file in PNG format.
≡	PngSaveImageToMemory (see page 239)	Saves image to the memory buffer in PNG format.

	PngSaveImageToStream (see page 240)	Saves image to the stream in PNG format.

Macros

Name	Description
PNG_DEFAULT_COMPRESSION_LEVEL (see page 241)	Specifies default PNG compression level.

Module

Png Module ([see page 237](#))

6.1.5.12 Tiff Module

Provides functionality for loading images in TIFF format.

Files

Name	Description
Tiff.h (see page 244)	Header file for Tiff module. Provides functionality for loading images in TIFF format.

Functions

	Name	Description
	TiffLoadImageFromFile (see page 242)	Loads image from TIFF file.
	TiffLoadImageFromMemory (see page 243)	Loads image from memory buffer containing TIFF file.
	TiffLoadImageFromStream (see page 243)	Loads image from stream containing TIFF file.

6.1.5.12.1 Functions

6.1.5.12.1.1 TiffLoadImageFromFile Function

Loads image from TIFF file.

C++

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

Parameters

Parameters	Description
const NChar * szFileName	[out] Points to string that specifies file name.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	the function fails, this error can be returned. This error code is returned when <i>szFileName</i> or <i>pHImage</i> is NULL (see page 133).

N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file specified by <i>szFileName</i> is invalid.

Remarks

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

Module

Tiff Module (see page 242)

6.1.5.12.1.2 TiffLoadImageFromMemory Function

Loads image from memory buffer containing TIFF file.

C++

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

Parameters

Parameters	Description
const void * buffer	[in] Pointer to memory buffer.
NSizeType bufferSize	[in] Length of memory buffer.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>buffer</i> is NULL (see page 133) and <i>bufferLength</i> is not equal to zero. - or - <i>pHImage</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file specified by <i>buffer</i> is invalid.

Remarks

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

Module

Tiff Module (see page 242)

6.1.5.12.1.3 TiffLoadImageFromStream Function

Loads image from stream containing TIFF file.

C++

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

Parameters

Parameters	Description
HNStream hStream	[in] Handle to HNStream containing Tiff image.

HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return values	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).

Module

Tiff Module (see page 242)

6.1.5.12.2 Files

6.1.5.12.2.1 Tiff.h

Header file for Tiff module. Provides functionality for loading images in TIFF format.

Functions

	Name	Description
≡	TiffLoadImageFromFile (see page 242)	Loads image from TIFF file.
≡	TiffLoadImageFromMemory (see page 243)	Loads image from memory buffer containing TIFF file.
≡	TiffLoadImageFromStream (see page 243)	Loads image from stream containing TIFF file.

Module

Tiff Module (see page 242)

6.1.5.13 NImageFormat Module

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

Files

Name	Description
NImageFormat.h (see page 259)	Header file for NImageFormat module. Provides functionality for loading and saving images in format-neutral way.

Functions

	Name	Description
≡	NImageFormatCanRead (see page 245)	Retrieves a value indicating whether the image format supports reading.
≡	NImageFormatCanWrite (see page 246)	Retrieves a value indicating whether the image format supports writing.
≡	NImageFormatCanWriteMultiple (see page 246)	Retrieves a value indicating whether the image format supports writing of multiple images.
≡	NImageFormatGetBmp (see page 247)	Retrieves BMP image format.
≡	NImageFormatGetDefaultFileExtension (see page 247)	Retrieves default extension of the image format.
≡	NImageFormatGetFileFilter (see page 248)	Retrieves file filter of the image format.

◆	NImageFormatGetFormat (see page 248)	Retrieves supported image format with the specified index.
◆	NImageFormatGetFormatCount (see page 249)	Retrieves number of supported image formats.
◆	NImageFormatGetIHead (see page 249)	Retrieves NIST IHead image format.
◆	NImageFormatGetJpeg (see page 249)	Retrieves JPEG image format.
◆	NImageFormatGetJpeg2K (see page 250)	Retrieves JPEG2K image format.
◆	NImageFormatGetName (see page 250)	Retrieves name of the image format.
◆	NImageFormatGetPng (see page 251)	Retrieves PNG image format.
◆	NImageFormatOpenFile (see page 251)	Opens image file with specified file name and image format.
◆	NImageFormatGetTiff (see page 252)	Retrieves TIFF image format.
◆	NImageFormatGetWsq (see page 252)	Retrieves WSQ image format.
◆	NImageFormatLoadImageFromFile (see page 253)	Loads image from file of specified image format.
◆	NImageFormatSaveImagesToFile (see page 253)	Saves specified number of images to the file in the specified format.
◆	NImageFormatLoadImageFromMemory (see page 254)	Loads image from the memory buffer containing file of the specified format.
◆	NImageFormatLoadImageFromStream (see page 254)	Loads image from the memory stream containing file of the specified format.
◆	NImageFormatOpenFileFromMemory (see page 255)	Opens image file from the memory buffer containing file of specified format.
◆	NImageFormatOpenFileFromStream (see page 256)	Opens image file from the memory stream containing file of specified format.
◆	NImageFormatSaveImageToFile (see page 256)	Saves image to the file in the specified format.
◆	NImageFormatSaveImageToMemory (see page 257)	Saves image to the memory buffer in the specified format.
◆	NImageFormatSaveImageToStream (see page 257)	Saves image to the memory stream in the specified format.
◆	NImageFormatSelect (see page 258)	Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.
	HNImageFormat (see page 259)	Handle to image format.

6.1.5.13.1 Functions

6.1.5.13.1.1 NImageFormatCanRead Function

Retrieves a value indicating whether the image format supports reading.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.

NBool * pValue	[out] Pointer to NBool (see page 116) that receives value indicating whether the image format supports reading.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> or <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.2 NImageFormatCanWrite Function

Retrieves a value indicating whether the image format supports writing.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
NBool * pValue	[out] Pointer to NBool (see page 116) that receives value indicating whether the image format supports writing.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> or <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.3 NImageFormatCanWriteMultiple Function

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

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
NBool * pValue	[out] Pointer to NBool (see page 116) that receives value indicating whether the image format supports writing of multiple images.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> or <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.4 NImageFormatGetBmp Function

Retrieves BMP image format.

C++

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

Parameters

Parameters	Description
HNImageFormat * pValue	[out] Pointer to HNImageFormat (see page 259) that receives handle to image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.5 NImageFormatGetDefaultFileExtension Function

Retrieves default extension of the image format.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
NChar * pValue	[out] Pointer to string that receives default file extension of the image format. Can be NULL (see page 133).

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
Length of a string	If the function succeeds and <i>pValue</i> is NULL (see page 133), the return value is length of the string (not including the NULL (see page 133)-terminator) <i>pValue</i> should point to.

N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> is NULL (see page 133).

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.6 NImageFormatGetFileFilter Function

Retrieves file filter of the image format.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
NChar * pValue	[out] Pointer to the string that receives file filter of the image format. Can be NULL (see page 133).

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
Length of a string	If the function succeeds and <i>pValue</i> is NULL (see page 133), the return value is length of the string (not including the NULL (see page 133)-terminator) <i>pValue</i> should point to.
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> is NULL (see page 133).

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.7 NImageFormatGetFormat Function

Retrieves supported image format with the specified index.

C++

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

Parameters

Parameters	Description
NInt index	[in] Specifies zero-based supported image format index to retrieve.
HNImageFormat * pValue	[out] Pointer to HNImageFormat (see page 259) that receives image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>index</i> is less than zero or greater than or equal to supported image format count.

N_E_ARGUMENT_NULL (see page 78)	see	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.8 NImageFormatGetFormatCount Function

Retrieves number of supported image formats.

C++

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

Parameters

Parameters	Description
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of supported image formats.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.9 NImageFormatGetIHead Function

Retrieves NIST IHead image format.

C++

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

Parameters

Parameters	Description
HNImageFormat * pValue	[out] Pointer to HNImageFormat (see page 259) that receives handle to image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.10 NImageFormatGetJpeg Function

Retrieves JPEG image format.

C++

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

Parameters

Parameters	Description
HNImageFormat * pValue	[out] Pointer to HNImageFormat (see page 259) that receives handle to image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.11 NImageFormatGetJpeg2K Function

Retrieves JPEG2K image format.

C++

```
NResult N_API NImageFormatGetJpeg2K(HNImageFormat * pValue);
```

Parameters

Parameters	Description
HNImageFormat * pValue	[out] Pointer to HNImageFormat (see page 259) that receives handle to image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.12 NImageFormatGetName Function

Retrieves name of the image format.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
NChar * pValue	[out] Pointer to the string that receives name of the image format. Can be NULL (see page 133).

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds and <i>pValue</i> is not NULL (see page 133), the return value is N_OK (see page 82).
Length of a string	If the function succeeds and <i>pValue</i> is NULL (see page 133), the return value is length of the string (not including the NULL (see page 133)-terminator) <i>pValue</i> should point to.
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> is NULL (see page 133).
Length of a string	If the function succeeds and <i>pValue</i> is NULL (see page 133), the return value is length of the string (not including the NULL (see page 133)-terminator) <i>pValue</i> should point to.

Module

NImageFormat Module (see page 244)

6.1.5.13.1.13 NImageFormatGetPng Function

Retrieves PNG image format.

C++

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

Parameters

Parameters	Description
HNImageFormat * pValue	[out] Pointer to HNImageFormat (see page 259) that receives handle to image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.14 NImageFormatOpenFile Function

Opens image file with specified file name and image format.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
const NChar * szFileName	[in] File name to open.
HNImageFile * pHImageFile	[out] Pointer to HNImageFile (see page 226) that receives handle to opened image file.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> or <i>pHImageFile</i> is NULL (see page 133).
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support reading.

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.15 NImageFormatGetTiff Function

Retrieves TIFF image format.

C++

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

Parameters

Parameters	Description
HNImageFormat * pValue	[out] Pointer to HNImageFormat (see page 259) that receives handle to image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.16 NImageFormatGetWsq Function

Retrieves WSQ image format.

C++

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

Parameters

Parameters	Description
HNImageFormat * pValue	[out] Pointer to HNImageFormat (see page 259) that receives handle to image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>pValue</i> is NULL (see page 133).

Module

NImageFormat Module (see page 244)

6.1.5.13.1.17 NImageFormatLoadImageFromFile Function

Loads image from file of specified image format.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
const NChar * szFileName	[in] Points to string that specifies file name.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> , <i>szFileName</i> or <i>pHImage</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file specified by <i>szFileName</i> is invalid.
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support reading.

Module

NImageFormat Module (see page 244)

6.1.5.13.1.18 NImageFormatSaveImagesToFile Function

Saves specified number of images to the file in the specified format.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
NInt imageCount	[in] Number of images to save.
HNImage * arHImages	[out] Handles to all images that should be saved.
const NChar * szFileName	[in] Points to memory buffer that contains file names for all images to save.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> or <i>arHImages</i> is NULL (see page 133).
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support writing.
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	If the function fails, this error can be returned. This error code is returned when image <i>imageCount</i> value is equal or less than zero.

Module

NImageFormat Module (see page 244)

6.1.5.13.1.19 NImageFormatLoadImageFromMemory Function

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

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
const void * buffer	[out] Pointer to memory buffer.
NSizeType bufferLength	[in] Length of memory buffer.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> or <i>pHImageFormat</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of the file contained in memory buffer specified by <i>buffer</i> is invalid for the specified image format.
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support reading.

Module

NImageFormat Module (see page 244)

6.1.5.13.1.20 NImageFormatLoadImageFromStream Function

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

C++

```
NResult N_API NImageFormatLoadImageFromStream(HNImageFormat hImageFormat, HNStream hStream,
```

```
HNImage * pHImage);
```

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
HNStream hStream	[in] Pointer to memory stream.
HNImage * pHImage	[out] Pointer to HNImage (see page 221) that receives handle to loaded image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of the file contained in memory stream specified by buffer is invalid for the specified image format.
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support reading.

Module

NImageFormat Module (see page 244)

6.1.5.13.1.21 NImageFormatOpenFileFromMemory Function

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

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
const void * buffer	[out] Pointer to memory buffer.
NSizeType bufferLength	[in] Length of memory buffer.
HNImageFile * pHImageFile	[out] Pointer to HNImageFile (see page 226) that receives handle to opened image file.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> or <i>pHImageFile</i> is NULL (see page 133) - or - <i>buffer</i> is NULL (see page 133) and <i>bufferLength</i> is not equal to zero.
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file contained in buffer specified by <i>buffer</i> is invalid for specified image format.

N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support reading.

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.22 NImageFormatOpenFileFromStream Function

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

C++

```
NResult N_API NImageFormatOpenFileFromStream(HNImageFormat hImageFormat, HNStream hStream,
NBool ownsStream, HNImageFile * pHImageFile);
```

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
HNStream hStream	[in] Handle to memory stream.
NBool ownsStream	[in] If true, the stream is closed by the writer when done; otherwise false.
HNImageFile * pHImageFile	[out] Pointer to HNImageFile (see page 226) that receives handle to opened image file.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> or <i>pHImageFile</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	If the function fails, this error can be returned. This error code is returned when format of file contained in buffer specified by buffer is invalid for specified image format.
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support reading.

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.23 NImageFormatSaveImageToFile Function

Saves image to the file in the specified format.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
HNImage hImage	[in] Handle to image.
const NChar * szFileName	[in] Points to string that specifies file name.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> , <i>hImage</i> (see page 142) or <i>szFileName</i> is NULL (see page 133).
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support writing.

Module

NImageFormat Module (see page 244)

6.1.5.13.1.24 NImageFormatSaveImageToMemory Function

Saves image to the memory buffer in the specified format.

C++

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

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
HNImage hImage	[in] Handle to image.
void * * pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
NSizeType * pBufferLength	[out] Pointer to NSizeType (see page 119) that receives size of allocated memory buffer.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> , <i>hImage</i> (see page 142), <i>pBuffer</i> or <i>pBufferLength</i> is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	If the function fails, this error can be returned. This error code is returned when there was not enough memory to allocate memory buffer.
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support writing.

Remarks

Memory buffer allocated by this function must be deallocated by NFree (see page 94) function when it is no longer needed.

Module

NImageFormat Module (see page 244)

6.1.5.13.1.25 NImageFormatSaveImageToStream Function

Saves image to the memory stream in the specified format.

C++

```
NResult N_API NImageFormatSaveImageToStream(HNImageFormat hImageFormat, HNImage hImage, HNStream hStream);
```

Parameters

Parameters	Description
HNImageFormat hImageFormat	[in] Handle to image format.
HNImage hImage	[in] Handle to image.
HNStream hStream	[in] Handle to memory stream that contains image.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>hImageFormat</i> , <i>hImage</i> (see page 142) or <i>hStream</i> is NULL (see page 133).
N_E_NOT_SUPPORTED (see page 81)	If the function fails, this error can be returned. This error code is returned when image format specified by <i>hImageFormat</i> does not support writing.

Module

NImageFormat Module (see page 244)

6.1.5.13.1.26 NImageFormatSelect Function

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

C++

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

Parameters

Parameters	Description
const NChar * szFileName	[in] Points to string that specifies file name.
NFileAccess fileAccess	[in] Specifies that image format should support reading, writing or both.
HNImageFormat * pHImageFormat	[out] Pointer to HNImageFormat (see page 259) that receives handle to image format.

Returns

Return value	Description
N_OK (see page 82)	If the function succeeds, the return value is N_OK (see page 82).
N_E_ARGUMENT (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>fileAccess</i> value is invalid.
N_E_ARGUMENT_NULL (see page 78)	If the function fails, this error can be returned. This error code is returned when <i>szFileName</i> or <i>pHImageFormat</i> is NULL (see page 133).

Remarks

If none of supported image formats that supports reading/writing as specified by *fileAccess* is registered with file extension of *szFileName* then handle returned via *pHImageFormat* is NULL (see page 133).

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.1.27 HNImageFormat

Handle to image format.

Module

NImageFormat Module ([see page 244](#))

6.1.5.13.2 Files**6.1.5.13.2.1 NImageFormat.h**

Header file for NImageFormat module. Provides functionality for loading and saving images in format-neutral way.

Functions

	Name	Description
	NImageFormatCanRead (see page 245)	Retrieves a value indicating whether the image format supports reading.
	NImageFormatCanWrite (see page 246)	Retrieves a value indicating whether the image format supports writing.
	NImageFormatCanWriteMultiple (see page 246)	Retrieves a value indicating whether the image format supports writing of multiple images.
	NImageFormatGetBmp (see page 247)	Retrieves BMP image format.
	NImageFormatGetDefaultFileExtension (see page 247)	Retrieves default extension of the image format.
	NImageFormatGetFileFilter (see page 248)	Retrieves file filter of the image format.
	NImageFormatGetFormat (see page 248)	Retrieves supported image format with the specified index.
	NImageFormatGetFormatCount (see page 249)	Retrieves number of supported image formats.
	NImageFormatGetIHead (see page 249)	Retrieves NIST IHead image format.
	NImageFormatGetJpeg (see page 249)	Retrieves JPEG image format.
	NImageFormatGetJpeg2K (see page 250)	Retrieves JPEG2K image format.
	NImageFormatGetName (see page 250)	Retrieves name of the image format.
	NImageFormatGetPng (see page 251)	Retrieves PNG image format.
	NImageFormatGetTiff (see page 252)	Retrieves TIFF image format.
	NImageFormatGetWsqa (see page 252)	Retrieves WSQ image format.
	NImageFormatLoadImageFromFile (see page 253)	Loads image from file of specified image format.
	NImageFormatLoadImageFromMemory (see page 254)	Loads image from the memory buffer containing file of the specified format.
	NImageFormatLoadImageFromStream (see page 254)	Loads image from the memory stream containing file of the specified format.

	NImageFormatOpenFile (see page 251)	Opens image file with specified file name and image format.
	NImageFormatOpenFileFromMemory (see page 255)	Opens image file from the memory buffer containing file of specified format.
	NImageFormatOpenFileFromStream (see page 256)	Opens image file from the memory stream containing file of specified format.
	NImageFormatSaveImagesToFile (see page 253)	Saves specified number of images to the file in the specified format.
	NImageFormatSaveImageToFile (see page 256)	Saves image to the file in the specified format.
	NImageFormatSaveImageToMemory (see page 257)	Saves image to the memory buffer in the specified format.
	NImageFormatSaveImageToStream (see page 257)	Saves image to the memory stream in the specified format.
	NImageFormatSelect (see page 258)	Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.

Module

NImageFormat Module ([see page 244](#))

6.1.6 NLicensing Library

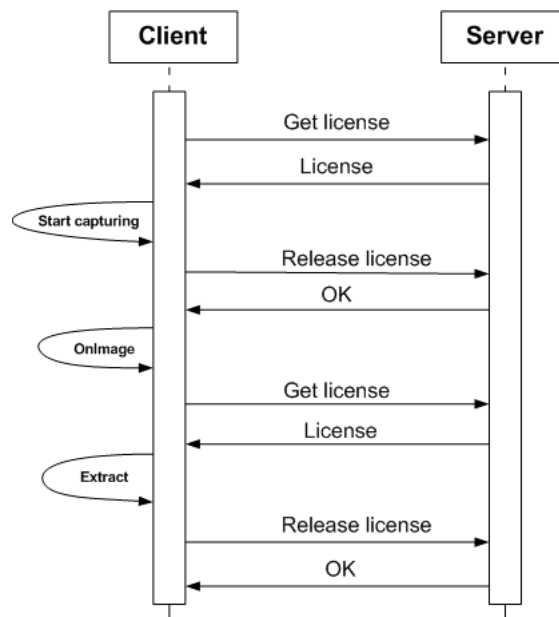
Provides functionality for getting, releasing licenses.

Remarks**Requirements (Windows)**

- **Import Library:** NLicensing.dll.lib
- **DLLs:**
 - NLicensing.dll
 - NCore.dll ([see page 74](#))

Requirements (Linux)

- **Shared objects:**
 - libNLicensing.so
 - libNcore.so

Concurrent license. Client and License Server scenario.**Modules**

Name	Description
NLicensing Module (see page 261)	Provides functionality for getting, releasing licenses.

6.1.6.1 NLicensing Module

Provides functionality for getting, releasing licenses.

Files

Name	Description
NLicensing.h (see page 267)	Header file for the NLicensing (see page 260) library. Provides functionality for getting and releasing licenses.

Functions

	Name	Description
≡	NLicenseGetInfo (see page 262)	Retrieves license information of specified product.
≡	NLicenseIsComponentActivated (see page 262)	Checks if component specified by name is activated.
≡	NLicenseObtain (see page 263)	Obtains product license from the specified server.
≡	NLicenseRelease (see page 265)	Releases license.
≡	NLicensingGetInfo (see page 266)	Gets information about the library.

Structs, Records, Enums

	Name	Description
📦	NLicenseInfo_ (see page 267)	Enumerates values describing license.
	NLicenseInfo (see page 267)	Enumerates values describing license.

6.1.6.1.1 Functions

6.1.6.1.1.1 NLicenseGetInfo Function

Retrieves license information of specified product.

C++

```
NResult N_API NLicenseGetInfo(const NChar * szProduct, NLicenseInfo * pLicenseInfo);
```

Parameters

Parameters	Description
const NChar * szProduct	[in] Zero terminated string which contains product name to retrieve license information.
NLicenseInfo * pLicenseInfo	[out] Pointer to NLicenseInfo (see page 267) structure.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	szProduct or pLicenseInfo is NULL (see page 133).

Module

NLicensing Module (see page 261)

6.1.6.1.1.2 NLicenseIsComponentActivated Function

Checks if component specified by name is activated.

C++

```
NResult N_API NLicenseIsComponentActivated(const NChar * szName, NBool * pValue);
```

Parameters

Parameters	Description
const NChar * szName	[in] Zero terminated string which contains component name.
NBool * pValue	[out] NBool (see page 116) value indicating if the component was activated.

Returns

If the function succeeds, the return value is N_OK (see page 82). If application working folder does not contain required files, N_E_FILE_NOT_FOUND error is thrown.

Remarks

These component names for szName parameter can be used:

- Matchers.Fusion
- Matchers.Fingers
- Matchers.Faces
- Matchers.Irises
- Matchers.Palms
- Matchers.Fingers Pro
- Matchers.Faces Pro
- Matchers.Irises Pro

- Matchers.Palms Pro
- Extractors.Fingers
- Extractors.Faces Detection
- Extractors.Faces Extraction
- Extractors.Irises
- Extractors.Palms

Module

NLicensing Module (see page 261)

6.1.6.1.1.3 NLicenseObtain Function

Obtains product license from the specified server.

C++

```
NResult N_API NLicenseObtain(const NChar * szAddress, const NChar * szPort, const NChar * szProducts, NBool * pAvailable);
```

Parameters

Parameters	Description
const NChar * szAddress	[in] Server address where license manager is installed as a server. License manager can be in the same or in other computer. /local used when license is tied with the computer using serial number.
const NChar * szPort	[in] License manager server port.
const NChar * szProducts	[in] Zero terminated string which contains license name. For a complete list of available values for this parameter see Remarks section.
NBool * pAvailable	[out] Returns NTrue (see page 133) if license available; otherwise, NFalse (see page 133).

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	szAddress, szPort, szProducts or pAvailable is NULL (see page 133).
N_E_FORMAT (see page 80)	szProducts incorrect format.
N_E_FAILED (see page 79)	Unspecified error has occurred.

Remarks

These values are available for szProducts parameter (by default concurrent license is obtained):

1. VeriFinger SDK:

1. SingleComputerLicense:VFExtractor or concurrent:VFExtractor
2. SingleComputerLicense:VFMatcher

VFExtractor - VeriFinger Extractor

VFMatcher - VeriFinger Matcher

2. VeriLook SDK:

1. SingleComputerLicense:VLExtractor or concurrent:VLExtractor
2. SingleComputerLicense:VLMatcher

VLExtractor - VeriLook Extractor

VLMatcher - VeriLook Matcher

3. VeriLook Surveillance SDK:

1. SingleComputerLicense:VLSurveillance

4. MegaMatcher SDK:

1. SingleComputerLicense:MegaMatcherServer
2. SingleComputerLicense:MegaMatcherClusterServer
3. SingleComputerLicense:MegaMatcherClusterNode
4. SingleComputerLicense:MegaMatcherClient
5. concurrent:MegaMatcherClient

5. VeriEye SDK:

1. SingleComputerLicense:VEExtractor
2. SingleComputerLicense:VEMatcher

VEExtractor - VeriEye Extractor

VEMatcher - VeriEye Matcher

6. BSS SDK:

1. SingleComputerLicense:VFBSS
2. SingleComputerLicense:VLBSS
3. SingleComputerLicense:VEBSS

7. VeriFinger VAR SDK:

1. SingleComputerLicense:VFExtractorV
2. SingleComputerLicense:VFMatcherV

VFExtractorV - VeriFinger Extractor

VFMatcherV - VeriFinger Matcher

8. VeriLook VAR SDK:

1. SingleComputerLicense:VLExtractorV
2. SingleComputerLicense:VLMatcherV

VLExtractorV - VeriLook Extractor

VLMatcherV - VeriLook Matcher

9. VeriEye VAR SDK:

1. SingleComputerLicense:VEExtractorV
2. SingleComputerLicense:VEMatcherV

VEExtractorV - VeriEye Extractor

VEMatcherV - VeriEye Matcher

10. SentiSight SDK:

1. SingleComputerLicense:SentiSight

Note: nonconcurrent licenses (SingleComputerLicense) **can not be released** with NLicenseRelease (see page 265) function, but .NLicenseRelease (see page 265) function must be called before program exits.

If concurrent license is in use, then `NLicenseRelease` (see page 265) function should be called after features extraction.

Notes

Acquired license must be released using `NLicenseRelease` (see page 265) function.

Module

NLicensing Module (see page 261)

6.1.6.1.1.4 NLicenseRelease Function

Releases license.

C++

```
NResult N_API NLicenseRelease(const NChar * szProducts);
```

Parameters

Parameters	Description
const NChar * szProducts	[in] Product names to release licenses.

Returns

If the function succeeds, the return value is `N_OK` (see page 82).

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

Error code	Condition
<code>N_E_ARGUMENT_NULL</code> (see page 78)	<code>szProducts</code> is NULL (see page 133).
<code>N_E_FORMAT</code> (see page 80)	<code>szProducts</code> incorrect format.
<code>N_E_FAILED</code> (see page 79)	Unspecified error has occurred.

Remarks

These values are available for `szProducts` parameter:

1. VeriFinger SDK:

- SingleComputerLicense:VFExtractor or concurrent:VFExtractor
- SingleComputerLicense:VFMatcher

VFExtractor - VeriFinger Extractor

VFMatcher - VeriFinger Matcher

2. VeriLook SDK:

- SingleComputerLicense:VLExtractor or concurrent:VLExtractor
- SingleComputerLicense:VLMatcher

VLExtractor - VeriLook Extractor

VLMatcher - VeriLook Matcher

3. VeriLook Surveillance SDK:

- SingleComputerLicense:VLSurveillance

4. MegaMatcher SDK:

- SingleComputerLicense:MegaMatcherServer
- SingleComputerLicense:MegaMatcherClusterServer
- SingleComputerLicense:MegaMatcherClusterNode

4. SingleComputerLicense:MegaMatcherClient
5. concurrent:MegaMatcherClient

5. VeriEye SDK:

1. SingleComputerLicense:VEExtractor
2. SingleComputerLicense:VEMatcher

VEExtractor - VeriEye Extractor

VEMatcher - VeriEye Matcher

6. BSS SDK:

1. SingleComputerLicense:VFBSS
2. SingleComputerLicense:VLBSS
3. SingleComputerLicense:VEBSS

7. VeriFinger VAR SDK:

1. SingleComputerLicense:VFExtractorV
2. SingleComputerLicense:VFMatcherV

VFExtractorV - VeriFinger Extractor

VFMatcherV - VeriFinger Matcher

8. VeriLook VAR SDK:

1. SingleComputerLicense:VLEExtractorV
2. SingleComputerLicense:VLMatcherV

VLEExtractorV - VeriLook Extractor

VLMatcherV - VeriLook Matcher

9. VeriEye VAR SDK:

1. SingleComputerLicense:VEExtractorV
2. SingleComputerLicense:VEMatcherV

VEExtractorV - VeriEye Extractor

VEMatcherV - VeriEye Matcher

10. SentiSight SDK:

1. SingleComputerLicense:SentiSight

Note: nonconcurrent licenses (SingleComputerLicense) **can not be released** with NLicenseRelease function, but .NLicenseRelease function must be called before program exits.

If concurrent license is in use, then NLicenseRelease function should be called after features extraction.

Module

NLicensing Module (see page 261)

6.1.6.1.1.5 NLicensingGetInfo Function

Gets information about the library.

C++

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

Parameters

Parameters	Description
NLibraryInfo * pValue	[out] Pointer to NLibraryInfo structure that receives library information.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pValue is NULL (see page 133).

Module

NLicensing Module ([see page 261](#))

6.1.6.1.2 Structs, Records, Enums

6.1.6.1.2.1 NLicenseInfo Structure

Enumerates values describing license.

C++

```
typedef struct NLicenseInfo_ {  
    NBool IsObtained;  
    NInt DistributorId;  
    NInt SerialNumber;  
} NLicenseInfo;
```

Members

Members	Description
NBool IsObtained;	Indicates if a license obtain was successful.
NInt DistributorId;	Id of a distributor which generated license.
NInt SerialNumber;	License serial number.

Module

NLicensing Module ([see page 261](#))

6.1.6.1.3 Files

6.1.6.1.3.1 NLicensing.h

Header file for the NLicensing ([see page 260](#)) library. Provides functionality for getting and releasing licenses.

Functions

	Name	Description
≡	NLicenseGetInfo (see page 262)	Retrieves license information of specified product.
≡	NLicenseIsComponentActivated (see page 262)	Checks if component specified by name is activated.
≡	NLicenseObtain (see page 263)	Obtains product license from the specified server.
≡	NLicenseRelease (see page 265)	Releases license.

	NLicensingGetInfo (see page 266)	Gets information about the library.

Module

NLicensing Module (see page 261)

Structures

	Name	Description
	NLicenseInfo_ (see page 267)	Enumerates values describing license.
	NLicenseInfo (see page 267)	Enumerates values describing license.

6.1.7 NSmartCards Library

This library provides functionality for working with ISO/IEC 7816-4:2005 compliant SIMPLE-TLV and interindustry BER-TLV data objects. It also contains modules that enable transmission of standardized ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005 compliant APDU commands (C-APDUs) to the card as well as retrieval of corresponding APDU responses (R-APDUs).

Modules

Name	Description
BerTlv Module (see page 268)	Provides functionality for creation, validation, retrieval, update and deletion of primitive and constructed BER-TLV data objects (BerTlv) that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.
NSmartCardsBiometry Module (see page 307)	Specifies ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 compliant types and subtypes of biometric data, challenge qualifiers also ISO/IEC 7816-11:2004 and ISO/IEC 7816-6:2004 compliant tag fields of interindustry BER-TLV data objects (BerTlv (see page 268)) used in ISO/IEC 7816-11:2004 Biometric Information Templates (BITs) and Biometric Data Templates (BDTs).
NSmartCardsCommands Module (see page 320)	Provides functionality for creation, validation, parsing (e.g., interindustry / proprietary format definition, etc.), transmission and inspection of ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005 compliant APDU commands (C-APDUs) and corresponding APDU responses (R-APDUs).
NSmartCardsDataElements Module (see page 459)	Specifies tags of ISO/IEC 7816-6:2004 interindustry data elements used for interchange.
SimpleTlv Module (see page 504)	Provides functionality for creation, validation, retrieval, update and deletion of SIMPLE-TLV data objects (SimpleTlv) that are fully compliant with SIMPLE-TLV encoding rules specified in ISO/IEC 7816-4:2005.

6.1.7.1 BerTlv Module

Provides functionality for creation, validation, retrieval, update and deletion of primitive and constructed BER-TLV data objects (BerTlv) that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Files

Name	Description
BerTlv.h (see page 303)	Provides functionality for creation, validation, retrieval, update and deletion of primitive and constructed BER-TLV data objects (BerTlv) that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Functions

	Name	Description
⇒	BerTagCreate (see page 272)	Creates BER-TLV tag field.
⇒	BerTagGetClass (see page 273)	Gets the class of the BER-TLV tag field.
⇒	BerTagGetEncoding (see page 273)	Gets the encoding of the BER-TLV data object.
⇒	BerTagGetLength (see page 273)	Gets the length of the BER-TLV tag field.
⇒	BerTagGetNumber (see page 274)	Gets the tag number encoded in the BER-TLV tag field.
⇒	BerTagsValid (see page 274)	Checks whether BER-TLV tag field is valid, i.e. does not violate any of the encoding rules specified in ISO/IEC 7816-4:2005.
⇒	BerTagRead (see page 274)	Reads BER-TLV tag field from memory buffer.
⇒	BerTagWrite (see page 275)	Writes BER-TLV tag field to memory buffer.
⇒	BerTlvClone (see page 275)	Creates a clone of a BER-TLV data object.
⇒	BerTlvCreate (see page 276)	Creates BER-TLV data object.
⇒	BerTlvCreateFromMemory (see page 276)	Creates BER-TLV data object from memory buffer.
⇒	BerTlvGetLength (see page 277)	Gets the BER-TLV data object length field.
⇒	BerTlvGetMinLengthSize (see page 277)	Gets the minimal size of the BER-TLV length field.
⇒	BerTlvGetSize (see page 278)	Gets the size of a BER-TLV data object.
⇒	BerTlvGetSizeMany (see page 278)	Gets the total size of an array of BER-TLV data objects.
⇒	BerTlvGetTag (see page 279)	Gets the BER-TLV data object tag field.
⇒	BerTlvLoadManyFromMemory (see page 279)	Creates an array of BER-TLV data objects from memory buffer.
⇒	BerTlvSaveManyToMemory (see page 280)	Writes an array of BER-TLV data objects to memory buffer.
⇒	BerTlvSaveToMemory (see page 280)	Writes BER-TLV data object to memory buffer.
⇒	BerTlvSetMinLengthSize (see page 281)	Sets the minimal size of the BER-TLV length field.
⇒	ConstructedBerTlvAddDataObject (see page 281)	Creates BER-TLV data object and encapsulates it in a constructed BER-TLV.
⇒	ConstructedBerTlvAddDataObjectCopy (see page 282)	Creates a clone of a BER-TLV data object and encapsulates it in a constructed BER-TLV.
⇒	ConstructedBerTlvAddDataObjectFromMemory (see page 282)	Reads BER-TLV data object from memory buffer and encapsulates it in a constructed BER-TLV.
⇒	ConstructedBerTlvClearDataObjects (see page 283)	Reads BER-TLV data object from memory buffer and encapsulates it in a constructed BER-TLV.
⇒	ConstructedBerTlvFindDataObject (see page 283)	Finds a BER-TLV data object with particular tag in the list of all BER-TLVs that are encapsulated in a constructed BER-TLV.
⇒	ConstructedBerTlvFindDataObjectIndex (see page 284)	Finds index of a BER-TLV data object with particular tag in the list of all BER-TLVs that are encapsulated in a constructed BER-TLV.
⇒	ConstructedBerTlvGetDataObject (see page 285)	Gets the BER-TLV data object that is encapsulated in a constructed BER-TLV.
⇒	ConstructedBerTlvGetDataObjectCapacity (see page 285)	Gets the capacity of all BER-TLV data objects that are encapsulated in a constructed BER-TLV.

ConstructedBerTlvGetDataObjectCount (see page 286)	Gets the number of BER-TLV data objects that are encapsulated in a constructed BER-TLV.
ConstructedBerTlvRemoveDataObject (see page 286)	Removes encapsulated BER-TLV data object from constructed BER-TLV and updates the size of the latter respectively.
ConstructedBerTlvSetDataObjectCapacity (see page 287)	Sets the capacity of all BER-TLV data objects that are encapsulated in a constructed BER-TLV.
PrimitiveBerTlvGetValue (see page 287)	Retrieves the value field of a primitive BER-TLV.
PrimitiveBerTlvGetValueAsByte (see page 287)	Retrieves the value field of a primitive BER-TLV as NByte (see page 116).
PrimitiveBerTlvGetValueAsBoolean (see page 288)	Retrieves the value field of a primitive BER-TLV as NBoolean (see page 116).
PrimitiveBerTlvGetValueAsInt16 (see page 288)	Retrieves the value field of a primitive BER-TLV as NInt16 (see page 117).
PrimitiveBerTlvGetValueAsInt32 (see page 289)	Retrieves the value field of a primitive BER-TLV as NInt32 (see page 118).
PrimitiveBerTlvGetValueAsInteger (see page 289)	Retrieves the value field of a primitive BER-TLV as integer. Can retrieve from 1 to 4 byte value.
PrimitiveBerTlvGetValueAsSByte (see page 290)	Retrieves the value field of a primitive BER-TLV as NSByte (see page 119).
PrimitiveBerTlvGetValueAsUInt16 (see page 290)	Retrieves the value field of a primitive BER-TLV as NUInt16 (see page 120).
PrimitiveBerTlvGetValueAsUInt32 (see page 291)	Retrieves the value field of a primitive BER-TLV as NUInt32 (see page 120).
PrimitiveBerTlvGetValuePtr (see page 291)	Gets the pointer to the value field of a primitive BER-TLV.
PrimitiveBerTlvSetLength (see page 292)	Sets the length field of a primitive BER-TLV and updates the size of the BER-TLV respectively.
PrimitiveBerTlvSetValue (see page 292)	Sets the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
PrimitiveBerTlvSetValueAsByte (see page 293)	Sets a NByte (see page 116) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
PrimitiveBerTlvSetValueAsBoolean (see page 293)	Sets a NBoolean (see page 116) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
PrimitiveBerTlvSetValueAsInt16 (see page 294)	Sets a NInt16 (see page 117) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
PrimitiveBerTlvSetValueAsInt32 (see page 294)	Sets a NInt32 (see page 118) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
PrimitiveBerTlvSetValueAsInteger (see page 295)	Sets an integer to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively. Can set from 1 to 4 byte value.
PrimitiveBerTlvSetValueAsSByte (see page 295)	Sets a NSByte (see page 119) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
PrimitiveBerTlvSetValueAsUInt16 (see page 296)	Sets an NUInt16 (see page 120) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
PrimitiveBerTlvSetValueAsUInt32 (see page 296)	Sets an NUInt32 (see page 120) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
HBerTlv (see page 297)	Handle to a BER-TLV data object.

Macros

Name	Description
BER_TAG_BIT_STRING (see page 297)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying bit string type.
BER_TAG_BMP_STRING (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying BMPString type.
BER_TAG_BOOLEAN (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying boolean type.
BER_TAG_CHARACTER_STRING (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying character string type.
BER_TAG_EMBEDDED_PDV (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying embedded PDV type.
BER_TAG_END_OF_CONTENTS (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying end of contents.
BER_TAG_ENUMERATED (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying enumerated type.
BER_TAG_EXTERNAL (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying external type.
BER_TAG_GENERAL_STRING (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GeneralString type.
BER_TAG_GENERALIZED_TIME (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GeneralizedTime type.
BER_TAG_GRAPHIC_STRING (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GraphicString type.
BER_TAG_IA5_STRING (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying IA5String type.
BER_TAG_INSTANCE_OF (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying instance-of type.
BER_TAG_INTEGER (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying integer type.
BER_TAG_ISO646_STRING (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying ISO646String type.
BER_TAG_NULL (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying null type.
BER_TAG_NUMERIC_STRING (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying NumericString type.
BER_TAG_OBJECT_DESCRIPTOR (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying ObjectDescriptor type.
BER_TAG_OBJECT_IDENTIFIER (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying object identifier type.
BER_TAG_OCTET_STRING (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying octet string type.
BER_TAG_PRINTABLE_STRING (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying PrintableString type.
BER_TAG_REAL (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying real type.
BER_TAG_RELATIVE_OBJECT_IDENTIFIER (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying relative object identifier type.
BER_TAG_SEQUENCE (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence type.
BER_TAG_SEQUENCE_OF (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence-of type.

BER_TAG_SET (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set type.
BER_TAG_SET_OF (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set-of type.
BER_TAG_T61_STRING (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying T61String type.
BER_TAG_TELETEX_STRING (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying TeletexString type.
BER_TAG_UNIVERSAL_STRING (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UniversalString type.
BER_TAG_UTC_TIME (see page 303)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTCTime type.
BER_TAG_UTF8_STRING (see page 303)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTF8String type.
BER_TAG_VIDEOTEX_STRING (see page 303)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying VideotexString type.
BER_TAG_VISIBLE_STRING (see page 303)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying VisibleString type.

Structs, Records, Enums

	Name	Description
	BerTagClass_ (see page 297)	Specifies a class of a BER-TLV data object.
	BerTagEncoding_ (see page 297)	Specifies an encoding of a BER-TLV data object.
	BerTagClass (see page 297)	Specifies a class of a BER-TLV data object.
	BerTagEncoding (see page 297)	Specifies an encoding of a BER-TLV data object.

6.1.7.1.1 Functions

6.1.7.1.1.1 BerTagCreate Function

Creates BER-TLV tag field.

C++

```
NResult N_API BerTagCreate(BerTagClass cls, BerTagEncoding encoding, NInt number, NInt * pTag);
```

Parameters

Parameters	Description
BerTagClass cls	[in] Class of the BER-TLV tag field to be created.
BerTagEncoding encoding	[in] Encoding of the BER-TLV tag field to be created.
NInt number	[in] Raw tag number of the BER-TLV tag field to be created.
NInt * pTag	[out] Pointer to NInt (see page 117) that receives created BER-TLV tag field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pTag is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	number is less than zero or is too big.

Module

BerTlv Module (see page 268)

6.1.7.1.1.2 BerTagGetClass Function

Gets the class of the BER-TLV tag field.

C++

```
BerTagClass N_API BerTagGetClass(NInt tag);
```

Parameters

Parameters	Description
NInt tag	[in] BER-TLV tag field Value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

BerTlv Module (see page 268)

6.1.7.1.1.3 BerTagGetEncoding Function

Gets the encoding of the BER-TLV data object.

C++

```
BerTagEncoding N_API BerTagGetEncoding(NInt tag);
```

Parameters

Parameters	Description
NInt tag	[in] BER-TLV tag field Value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

BerTlv Module (see page 268)

6.1.7.1.1.4 BerTagGetLength Function

Gets the length of the BER-TLV tag field.

C++

```
NSizeType N_API BerTagGetLength(NInt tag);
```

Parameters

Parameters	Description
NInt tag	[in] BER-TLV tag field value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

BerTlv Module (see page 268)

6.1.7.1.1.5 BerTagGetNumber Function

Gets the tag number encoded in the BER-TLV tag field.

C++

```
NInt N_API BerTagGetNumber(NInt tag);
```

Parameters

Parameters	Description
NInt tag	[in] BER-TLV tag field value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

BerTlv Module (see page 268)

6.1.7.1.1.6 BerTagIsValid Function

Checks whether BER-TLV tag field is valid, i.e. does not violate any of the encoding rules specified in ISO/IEC 7816-4:2005.

C++

```
NBool N_API BerTagIsValid(NInt tag, NBool ffIsValidForTheFirstByte);
```

Parameters

Parameters	Description
NInt tag	[in] BER-TLV tag field value.
NBool ffIsValidForTheFirstByte	[in] Boolean parameter specifying whether 0xFF value is valid for the first byte of BER-TLV tag fields.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

BerTlv Module (see page 268)

6.1.7.1.1.7 BerTagRead Function

Reads BER-TLV tag field from memory buffer.

C++

```
NResult N_API BerTagRead(const void * pBuffer, NSizeType bufferSize, NInt * pTag);
```

Parameters

Parameters	Description
const void * pBuffer	[in] Pointer to memory buffer that contains BER-TLV tag field.
NSizeType bufferSize	[in] Size of memory buffer that contains BER-TLV tag field.
NInt * pTag	[out] Pointer to NInt (see page 117) that receives read BER-TLV tag field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pBuffer or pTag is NULL (see page 133).

Module

BerTlv Module (see page 268)

6.1.7.1.1.8 BerTagWrite Function

Writes BER-TLV tag field to memory buffer.

C++

```
NResult N_API BerTagWrite(NInt tag, void * pBuffer, NSizeType bufferSize, NSizeType * pSize);
```

Parameters

Parameters	Description
NInt tag	[in] BER-TLV tag field value.
void * pBuffer	[out] Pointer to memory buffer to write BER-TLV tag field in.
NSizeType bufferSize	[in] Size of memory buffer to write BER-TLV tag field in.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives size of written BER-TLV tag field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pBuffer or pSize is NULL (see page 133).

Module

BerTlv Module (see page 268)

6.1.7.1.1.9 BerTlvClone Function

Creates a clone of a BER-TLV data object.

C++

```
NResult N_API BerTlvClone(HBerTlv hBerTlv, HBerTlv * pHClonedBerTlv);
```

Parameters

Parameters	Description
HBerTlv hBerTlv	[in] Handle to a BerTlv (see page 268) object to be cloned.
HBerTlv * pHClonedBerTlv	[out] Pointer to HBerTlv (see page 297) that receives a handle to the clone of a BerTlv (see page 268) object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hBerTlv or pHClonedBerTlv is NULL (see page 133).

Module

BerTlv Module ([see page 268](#))

6.1.7.1.1.10 BerTlvCreate Function

Creates BER-TLV data object.

C++

```
NResult N_API BerTlvCreate(NInt tag, NInt minLengthSize, NBool ffIsValidForTheTagFirstByte,
HBerTlv * pHBerTlv);
```

Parameters

Parameters	Description
NInt tag	[in] BER-TLV tag field value.
NInt minLengthSize	[in] Minimal size of the the BER-TLV length field.
NBool ffIsValidForTheTagFirstByte	[in] Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.
HBerTlv * pHBerTlv	[out] Pointer to HBerTlv (see page 297) that receives a handle to created BER-TLV object.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT (see page 78)	If tag field is valid or minLengthSize is out of the range [1,5].
N_E_ARGUMENT_NULL (see page 78)	pHBerTlv is NULL (see page 133).

Module

BerTlv Module ([see page 268](#))

6.1.7.1.1.11 BerTlvCreateFromMemory Function

Creates BER-TLV data object from memory buffer.

C++

```
NResult N_API BerTlvCreateFromMemory(const void * pBuffer, NSizeType bufferSize, NBool
ffIsValidForTheTagFirstByte, HBerTlv * pHBerTlv);
```

Parameters

Parameters	Description
const void * pBuffer	[in] Pointer to memory buffer to read BER-TLV data from.
NSizeType bufferSize	Size of memory buffer to read BER-TLV data from.
NBool ffIsValidForTheTagFirstByte	[in] Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.
HBerTlv * pHBerTlv	[out] Pointer to HBerTlv (see page 297) that receives a handle to created BER-TLV data object.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pBuffer</i> or <i>pHBerTlv</i> is NULL (see page 133).

Module

BerTlv Module (see page 268)

6.1.7.1.1.12 BerTlvGetLength Function

Gets the BER-TLV data object length field.

C++

```
NResult N_API BerTlvGetLength(HBerTlv hBerTlv, NSizeType * pValue);
```

Parameters

Parameters	Description
HBerTlv hBerTlv	[in] Handle to a BER-TLV data object.
NSizeType * pValue	[out] Pointer to NInt (see page 117) that receives the value of the BER-TLV length field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).

Module

BerTlv Module (see page 268)

6.1.7.1.1.13 BerTlvGetMinLengthSize Function

Gets the minimal size of the BER-TLV length field.

C++

```
NResult N_API BerTlvGetMinLengthSize(HBerTlv hBerTlv, NInt * pValue);
```

Parameters

Parameters	Description
HBerTlv hBerTlv	[in] Handle to a BER-TLV data object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives the minimal size of the BER-TLV data object length field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).

Module

BerTlv Module ([see page 268](#))

6.1.7.1.1.14 BerTlvGetSize Function

Gets the size of a BER-TLV data object.

C++

```
NResult N_API BerTlvGetSize(HBerTlv hBerTlv, NSizeType * pValue);
```

Parameters

Parameters	Description
HBerTlv hBerTlv	[in] Handle to a BER-TLV data object.
NSizeType * pValue	[out] Pointer to NSizeType (see page 119) that receives the value of the BER-TLV size.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).

Module

BerTlv Module ([see page 268](#))

6.1.7.1.1.15 BerTlvGetSizeMany Function

Gets the total size of an array of BER-TLV data objects.

C++

```
NResult N_API BerTlvGetSizeMany(const HBerTlv * arHBerTlvs, NInt count, NSizeType * pSize);
```

Parameters

Parameters	Description
const HBerTlv * arHBerTlvs	[in] Array of handles to BER-TLV data objects.
NInt count	[in] Number of elements in the array.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives the total size of all BER-TLV data objects from the array.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>arHBerTlvs</i> is NULL (see page 133) and count not equal zero.
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>count</i> is less than zero.

Module

BerTlv Module ([see page 268](#))

6.1.7.1.1.16 BerTlvGetTag Function

Gets the BER-TLV data object tag field.

C++

```
NResult N_API BerTlvGetTag(HBerTlv hBerTlv, NInt * pValue);
```

Parameters

Parameters	Description
HBerTlv hBerTlv	[in] Handle to a BER-TLV data object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives the value of the BER-TLV tag field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).

Module

BerTlv Module (see page 268)

6.1.7.1.1.17 BerTlvLoadManyFromMemory Function

Creates an array of BER-TLV data objects from memory buffer.

C++

```
NResult N_API BerTlvLoadManyFromMemory(const void * pBuffer, NSizeType bufferSize, NBool ffIsValidForTheTagFirstByte, HBerTlv * * pArHBerTlvs, NInt * pCount);
```

Parameters

Parameters	Description
const void * pBuffer	[in] Pointer to memory buffer to read BER-TLVs data from.
NSizeType bufferSize	Size of memory buffer to read BER-TLVs data from.
NBool ffIsValidForTheTagFirstByte	[in] Boolean parameter specifying whether 0xFF value is valid for the first byte of BER-TLV tag fields.
HBerTlv * * pArHBerTlvs	[out] Pointer to an array of pointers to handles of created BER-TLV data objects.
NInt * pCount	[out] Pointer to NInt (see page 117) that receives the number of created BER-TLV data objects.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pBuffer</i> is NULL (see page 133) and <i>bufferSize</i> is not zero. <i>pArHBerTlvs</i> or <i>pCount</i> IS NULL (see page 133).

Module

BerTlv Module (see page 268)

6.1.7.1.1.18 BerTlvSaveManyToMemory Function

Writes an array of BER-TLV data objects to memory buffer.

C++

```
NResult N_API BerTlvSaveManyToMemory(const HBerTlv * arHBerTlvs, NInt count, void * pBuffer, NSizeType bufferSize, NSizeType * pSize);
```

Parameters

Parameters	Description
const HBerTlv * arHBerTlvs	[in] Pointer to an array of handles to BER-TLV data objects.
NInt count	[in] Number of elements in the array.
void * pBuffer	[out] Pointer to memory buffer to write BER-TLVs in.
NSizeType bufferSize	Size of memory buffer to write BER-TLVs in.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives total size of written BER-TLVs.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>bufferSize</i> is not big enough.
N_E_ARGUMENT_NULL (see page 78)	<i>pBuffer</i> is NULL (see page 133) and <i>bufferSize</i> is not zero.

Module

BerTlv Module (see page 268)

6.1.7.1.1.19 BerTlvSaveToMemory Function

Writes BER-TLV data object to memory buffer.

C++

```
NResult N_API BerTlvSaveToMemory(HBerTlv hBerTlv, void * pBuffer, NSizeType bufferSize, NSizeType * pSize);
```

Parameters

Parameters	Description
HBerTlv hBerTlv	[in] Handle to a BER-TLV data object to be written.
void * pBuffer	[out] Pointer to memory buffer to write BER-TLV in.
NSizeType bufferSize	Size of memory buffer to write BER-TLV in.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives the size of written BER-TLV data object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>bufferSize</i> is not big enough.

N_E_ARGUMENT_NULL see page 78)	(?) <i>hBerTlv</i> or <i>pBuffer</i> , or <i>pSize</i> is NULL (? see page 133).

ModuleBerTlv Module ([?](#) see page 268)**6.1.7.1.1.20 BerTlvSetMinLengthSize Function**

Sets the minimal size of the BER-TLV length field.

C++

```
NResult N_API BerTlvSetMinLengthSize(HBerTlv hBerTlv, NInt value);
```

Parameters

Parameters	Description
HBerTlv hBerTlv	[in] Handle to a BER-TLV data object.
NInt value	[in] The minimal size of the BER-TLV length field to be set.

Returns

If the function succeeds, the return value is N_OK ([?](#) see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (? see page 78)	<i>hBerTlv</i> is NULL (? see page 133).

ModuleBerTlv Module ([?](#) see page 268)**6.1.7.1.1.21 ConstructedBerTlvAddDataObject Function**

Creates BER-TLV data object and encapsulates it in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvAddDataObject(HConstructedBerTlv hBerTlv, NInt tag, NInt minLengthSize, NBool ffIsValidForTheTagFirstByte, HBerTlv * pHDataObject);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
NInt tag	[in] Tag field value of the BER-TLV data object to be encapsulated.
NInt minLengthSize	[in] Minimal length size of the BER-TLV data object to be encapsulated.
NBool ffIsValidForTheTagFirstByte	[in] Boolean parameter specifying whether 0xFF value is valid for the first byte of the tag field of the BER-TLV to be encapsulated.
HBerTlv * pHDataObject	[out] Pointer to HBerTlv (? see page 297) that receives a handle to the encapsulated BER-TLV data object.

Returns

If the function succeeds, the return value is N_OK ([?](#) see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pHDataObject</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.22 ConstructedBerTlvAddDataObjectCopy Function

Creates a clone of a BER-TLV data object and encapsulates it in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvAddDataObjectCopy(HConstructedBerTlv hBerTlv, HBerTlv hSrcDataObject, HBerTlv * pHDataObject);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
HBerTlv hSrcDataObject	[in] Handle to the BER-TLV data object to be cloned.
HBerTlv * pHDataObject	[out] Pointer to HBerTlv (see page 297) that receives a handle to the encapsulated BER-TLV clone.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>hSrcDataObject</i> , or <i>pHDataObject</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.23 ConstructedBerTlvAddDataObjectFromMemory Function

Reads BER-TLV data object from memory buffer and encapsulates it in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvAddDataObjectFromMemory(HConstructedBerTlv hBerTlv, const void * pBuffer, NSizeType bufferSize, NBool ffIsValidForTheTagFirstByte, HBerTlv * pHDataObject);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
const void * pBuffer	[in] Pointer to memory buffer to read BER-TLV data from.
NSizeType bufferSize	Size of memory buffer to read BER-TLV data from.
NBool ffIsValidForTheTagFirstByte	[in] Boolean parameter specifying whether 0xFF value is valid for the first byte of the tag field of the BER-TLV to be encapsulated.

HBerTlv * pHDataObject	[out] Pointer to HBerTlv (see page 297) that receives a handle to the encapsulated BER-TLV data object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hBerTlv or pHDataObject is NULL (see page 133).
N_E_FORMAT (see page 80)	pBerTlv is not nbteConstructed.

Module

BerTlv Module (see page 268)

6.1.7.1.1.24 ConstructedBerTlvClearDataObjects Function

Reads BER-TLV data object from memory buffer and encapsulates it in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvClearDataObjects(HConstructedBerTlv hBerTlv);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
pBuffer	[in] Pointer to memory buffer to read BER-TLV data from.
bufferSize	Size of memory buffer to read BER-TLV data from.
ffIsValidForTheTagFirstByte	[in] Boolean parameter specifying whether 0xFF value is valid for the first byte of the tag field of the BER-TLV to be encapsulated.
pHDataObject	[out] Pointer to HBerTlv (see page 297) that receives a handle to the encapsulated BER-TLV data object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hBerTlv or pHDataObject is NULL (see page 133).
N_E_FORMAT (see page 80)	pBerTlv is not nbteConstructed.

Module

BerTlv Module (see page 268)

6.1.7.1.1.25 ConstructedBerTlvFindDataObject Function

Finds a BER-TLV data object with particular tag in the list of all BER-TLVs that are encapsulated in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvFindDataObject(HConstructedBerTlv hBerTlv, NByte tag,
HBerTlv * pHDataObject);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
NByte tag	[in] Tag field value of the encapsulated BER-TLV data object to be searched.
HBerTlv * pHDataObject	[out] Pointer to HBerTlv (see page 297) that receives a handle to the encapsulated BER-TLV in case of successful search or NULL (see page 133) otherwise.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pHDataObject</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.26 ConstructedBerTlvFindDataObjectIndex Function

Finds index of a BER-TLV data object with particular tag in the list of all BER-TLVs that are encapsulated in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvFindDataObjectIndex(HConstructedBerTlv hBerTlv, NByte tag, NInt * pIndex);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
NByte tag	[in] Tag field value of the encapsulated BER-TLV data object to be searched.
NInt * pIndex	[out] Pointer to NInt (see page 117) that receives an index of the encapsulated BER-TLV in case of successful search or -1 otherwise.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pIndex</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.27 ConstructedBerTlvGetDataObject Function

Gets the BER-TLV data object that is encapsulated in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvGetDataObject(HConstructedBerTlv hBerTlv, NInt index,
HBerTlv * pHValue);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
NInt index	[in] Index of the encapsulated BER-TLV data object.
HBerTlv * pHValue	[out] Pointer to HBerTlv (see page 297) that receives a handle to the encapsulated BER-TLV data object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pHValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.28 ConstructedBerTlvGetDataObjectCapacity Function

Gets the capacity of all BER-TLV data objects that are encapsulated in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvGetDataObjectCapacity(HConstructedBerTlv hBerTlv, NInt *
pValue);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives the capacity of all BER-TLVs that are encapsulated in a constructed BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.29 ConstructedBerTlvGetDataObjectCount Function

Gets the number of BER-TLV data objects that are encapsulated in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvGetDataObjectCount(HConstructedBerTlv hBerTlv, NInt *  
pValue);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives the number of BER-TLV data objects that are encapsulated in a constructed BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133) or <i>pValue</i> is NULL (see page 133) and value length is not equal 0.
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.30 ConstructedBerTlvRemoveDataObject Function

Removes encapsulated BER-TLV data object from constructed BER-TLV and updates the size of the latter respectively.

C++

```
NResult N_API ConstructedBerTlvRemoveDataObject(HConstructedBerTlv hBerTlv, NInt index);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	[in] Handle to a constructed BER-TLV data object.
NInt index	[in] Index of the encapsulated BER-TLV data object to be removed.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.31 ConstructedBerTlvSetDataObjectCapacity Function

Sets the capacity of all BER-TLV data objects that are encapsulated in a constructed BER-TLV.

C++

```
NResult N_API ConstructedBerTlvSetDataObjectCapacity(HConstructedBerTlv hBerTlv, NInt value);
```

Parameters

Parameters	Description
HConstructedBerTlv hBerTlv	Handle to a constructed BER-TLV data object.
NInt value	The capacity of encapsulated BER-TLVs to be set.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbteConstructed</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.32 PrimitiveBerTlvGetValue Function

Retrieves the value field of a primitive BER-TLV.

C++

```
NResult N_API PrimitiveBerTlvGetValue(HPrimitiveBerTlv hBerTlv, void * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
void * pValue	[out] Pointer to void that receives the copy of the value field of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133) or <i>pValue</i> is NULL (see page 133) and <i>hBerTlv</i> length is not zero.
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.33 PrimitiveBerTlvGetValueAsByte Function

Retrieves the value field of a primitive BER-TLV as NByte (see page 116).

C++

```
NResult N_API PrimitiveBerTlvGetValueAsByte(HPrimitiveBerTlv hBerTlv, NByte * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NByte * pValue	[out] Pointer to NByte (see page 116) that receives the field value of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbtePrimitive</i> or if value length is not equal to size of NByte (see page 116).

Module

BerTlv Module (see page 268)

6.1.7.1.1.34 PrimitiveBerTlvGetValueAsBoolean Function

Retrieves the value field of a primitive BER-TLV as NBoolean (see page 116).

C++

```
NResult N_API PrimitiveBerTlvGetValueAsBoolean(HPrimitiveBerTlv hBerTlv, NBool * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NBool * pValue	[out] Pointer to NBool (see page 116) that receives the field value of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbtePrimitive</i> or if value length is not equal to 1.

Module

BerTlv Module (see page 268)

6.1.7.1.1.35 PrimitiveBerTlvGetValueAsInt16 Function

Retrieves the value field of a primitive BER-TLV as NInt16 (see page 117).

C++

```
NResult N_API PrimitiveBerTlvGetValueAsInt16(HPrimitiveBerTlv hBerTlv, NShort * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NShort * pValue	[out] Pointer to NShort (see page 119) that receives the field value of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> or if value length is not equal to size of NShort (see page 119).

Module

BerTlv Module (see page 268)

6.1.7.1.1.36 PrimitiveBerTlvGetValueAsInt32 Function

Retrieves the value field of a primitive BER-TLV as NInt32 (see page 118).

C++

```
NResult N_API PrimitiveBerTlvGetValueAsInt32(HPrimitiveBerTlv hBerTlv, NInt * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives the field value of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.37 PrimitiveBerTlvGetValueAsInteger Function

Retrieves the value field of a primitive BER-TLV as integer. Can retrieve from 1 to 4 byte value.

C++

```
NResult N_API PrimitiveBerTlvGetValueAsInteger(HPrimitiveBerTlv hBerTlv, NInt * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives the field value of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> or if value length is equal zero or greater than 4.

Module

BerTlv Module (see page 268)

6.1.7.1.1.38 PrimitiveBerTlvGetValueAsSByte Function

Retrieves the value field of a primitive BER-TLV as NSByte (see page 119).

C++

```
NResult N_API PrimitiveBerTlvGetValueAsSByte(HPrimitiveBerTlv hBerTlv, NSByte * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NSByte * pValue	[out] Pointer to NSByte (see page 119) that receives the field value of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> or if value length is not equal to size of NSByte (see page 119).

Module

BerTlv Module (see page 268)

6.1.7.1.1.39 PrimitiveBerTlvGetValueAsUInt16 Function

Retrieves the value field of a primitive BER-TLV as NUInt16 (see page 120).

C++

```
NResult N_API PrimitiveBerTlvGetValueAsUInt16(HPrimitiveBerTlv hBerTlv, NUShort * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives the field value of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> or if value length is not equal to size of NShort (see page 119).

Module

BerTlv Module (see page 268)

6.1.7.1.1.40 PrimitiveBerTlvGetValueAsUInt32 Function

Retrieves the value field of a primitive BER-TLV as NUInt32 (see page 120).

C++

```
NResult N_API PrimitiveBerTlvGetValueAsUInt32(HPrimitiveBerTlv hBerTlv, NUInt * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NUInt * pValue	[out] Pointer to NUInt (see page 119) that receives the field value of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> or <i>pValue</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> or if value length is not equal to size of NUInt (see page 119).

Module

BerTlv Module (see page 268)

6.1.7.1.1.41 PrimitiveBerTlvGetValuePtr Function

Gets the pointer to the value field of a primitive BER-TLV.

C++

```
NResult N_API PrimitiveBerTlvGetValuePtr(HPrimitiveBerTlv hBerTlv, void * * pValue);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
void ** pValue	[out] Pointer to void * that receives the pointer to the value field of a primitive BER-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hBerTlv or pValue is NULL (see page 133).
N_E_FORMAT (see page 80)	hBerTlv is not nbtePrimitive.

Module

BerTlv Module (see page 268)

6.1.7.1.1.42 PrimitiveBerTlvSetLength Function

Sets the length field of a primitive BER-TLV and updates the size of the BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetLength(HPrimitiveBerTlv hBerTlv, NSizeType value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NSizeType value	[in] BER-TLV length field value to be set.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hBerTlv is NULL (see page 133).
N_E_FORMAT (see page 80)	hBerTlv is not nbtePrimitive.

Module

BerTlv Module (see page 268)

6.1.7.1.1.43 PrimitiveBerTlvSetValue Function

Sets the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetValue(HPrimitiveBerTlv hBerTlv, const void * pValue, NSizeType length);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.

const void * pValue	[in] Pointer to the value to be set to the primitive BER-TLV value field.
NSizeType length	[in] Length of the value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hBerTlv or pValue is NULL (see page 133) or if value length is not equal zero.
N_E_FORMAT (see page 80)	hBerTlv is not nbtePrimitive.

Module

BerTlv Module (see page 268)

6.1.7.1.1.44 PrimitiveBerTlvSetValueAsByte Function

Sets a NByte (see page 116) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetValueAsByte(HPrimitiveBerTlv hBerTlv, NByte value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NByte value	[in] Unsigned byte value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hBerTlv is NULL (see page 133).
N_E_FORMAT (see page 80)	hBerTlv is not nbtePrimitive.

Module

BerTlv Module (see page 268)

6.1.7.1.1.45 PrimitiveBerTlvSetValueAsBoolean Function

Sets a NBoolean (see page 116) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetValueAsBoolean(HPrimitiveBerTlv hBerTlv, NBool value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.

NBool value	[in] Boolean value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.46 PrimitiveBerTlvSetValueAsInt16 Function

Sets a NInt16 (see page 117) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetValueAsInt16(HPrimitiveBerTlv hBerTlv, NShort value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NShort value	[in] Short value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.47 PrimitiveBerTlvSetValueAsInt32 Function

Sets a NInt32 (see page 118) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetValueAsInt32(HPrimitiveBerTlv hBerTlv, NInt value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NInt value	[in] Integer value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.48 PrimitiveBerTlvSetValueAsInteger Function

Sets an integer to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively. Can set from 1 to 4 byte value.

C++

```
NResult N_API PrimitiveBerTlvSetValueAsInteger(HPrimitiveBerTlv hBerTlv, NInt value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NInt value	[in] Integer value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.49 PrimitiveBerTlvSetValueAsSByte Function

Sets a NSByte (see page 119) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetValueAsSByte(HPrimitiveBerTlv hBerTlv, NSByte value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NSByte value	[in] Byte value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the

following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.50 PrimitiveBerTlvSetValueAsUInt16 Function

Sets an NUInt16 (see page 120) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetValueAsUInt16(HPrimitiveBerTlv hBerTlv, NUShort value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NUShort value	[in] Unsigned short value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>pBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.51 PrimitiveBerTlvSetValueAsUInt32 Function

Sets an NUInt32 (see page 120) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

C++

```
NResult N_API PrimitiveBerTlvSetValueAsUInt32(HPrimitiveBerTlv hBerTlv, NUInt value);
```

Parameters

Parameters	Description
HPrimitiveBerTlv hBerTlv	[in] Handle to a primitive BER-TLV data object.
NUInt value	[in] Unsigned integer value to be set to the primitive BER-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hBerTlv</i> is NULL (see page 133).
N_E_FORMAT (see page 80)	<i>hBerTlv</i> is not <i>nbtePrimitive</i> .

Module

BerTlv Module (see page 268)

6.1.7.1.1.52 HBerTlv

Handle to a BER-TLV data object.

Module

BerTlv Module (see page 268)

6.1.7.1.2 Structs, Records, Enums

6.1.7.1.2.1 BerTagClass Enumeration

Specifies a class of a BER-TLV data object.

C++

```
typedef enum BerTagClass_ {  
} BerTagClass;
```

Module

BerTlv Module (see page 268)

6.1.7.1.2.2 BerTagEncoding Enumeration

Specifies an encoding of a BER-TLV data object.

C++

```
typedef enum BerTagEncoding_ {  
} BerTagEncoding;
```

Module

BerTlv Module (see page 268)

6.1.7.1.3 Macros

6.1.7.1.3.1 BER_TAG_BIT_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying bit string type.

C++

```
#define BER_TAG_BIT_STRING 3
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.2 BER_TAG_BMP_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying BMPString type.

C++

```
#define BER_TAG_BMP_STRING 30
```

Module

BerTlv Module ([see page 268](#))

6.1.7.1.3.3 BER_TAG_BOOLEAN Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying boolean type.

C++

```
#define BER_TAG_BOOLEAN 1
```

Module

BerTlv Module ([see page 268](#))

6.1.7.1.3.4 BER_TAG_CHARACTER_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying character string type.

C++

```
#define BER_TAG_CHARACTER_STRING 29
```

Module

BerTlv Module ([see page 268](#))

6.1.7.1.3.5 BER_TAG_EMBEDDED_PDV Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying embedded PDV type.

C++

```
#define BER_TAG_EMBEDDED_PDV 11
```

Module

BerTlv Module ([see page 268](#))

6.1.7.1.3.6 BER_TAG_END_OF_CONTENTS Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying end of contents.

C++

```
#define BER_TAG_END_OF_CONTENTS 0
```

Module

BerTlv Module ([see page 268](#))

6.1.7.1.3.7 BER_TAG_ENUMERATED Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying enumerated type.

C++

```
#define BER_TAG_ENUMERATED 10
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.8 BER_TAG_EXTERNAL Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying external type.

C++

```
#define BER_TAG_EXTERNAL 8
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.9 BER_TAG_GENERAL_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GeneralString type.

C++

```
#define BER_TAG_GENERAL_STRING 27
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.10 BER_TAG_GENERALIZED_TIME Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GeneralizedTime type.

C++

```
#define BER_TAG_GENERALIZED_TIME 24
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.11 BER_TAG_GRAPHIC_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GraphicString type.

C++

```
#define BER_TAG_GRAPHIC_STRING 25
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.12 BER_TAG_IA5_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying IA5String type.

C++

```
#define BER_TAG_IA5_STRING 22
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.13 BER_TAG_INSTANCE_OF Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying instance-of type.

C++

```
#define BER_TAG_INSTANCE_OF 8
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.14 BER_TAG_INTEGER Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying integer type.

C++

```
#define BER_TAG_INTEGER 2
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.15 BER_TAG_ISO646_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying ISO646String type.

C++

```
#define BER_TAG_ISO646_STRING 26
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.16 BER_TAG_NULL Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying null type.

C++

```
#define BER_TAG_NULL 5
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.17 BER_TAG_NUMERIC_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying NumericString type.

C++

```
#define BER_TAG_NUMERIC_STRING 18
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.18 BER_TAG_OBJECT_DESCRIPTOR Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying ObjectDescriptor type.

C++

```
#define BER_TAG_OBJECT_DESCRIPTOR 7
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.19 BER_TAG_OBJECT_IDENTIFIER Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying object identifier type.

C++

```
#define BER_TAG_OBJECT_IDENTIFIER 6
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.20 BER_TAG_OCTET_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying octet string type.

C++

```
#define BER_TAG_OCTET_STRING 4
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.21 BER_TAG_PRINTABLE_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying PrintableString type.

C++

```
#define BER_TAG_PRINTABLE_STRING 19
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.22 BER_TAG_REAL Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying real type.

C++

```
#define BER_TAG_REAL 9
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.23 BER_TAG_RELATIVE_OBJECT_IDENTIFIER Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying relative object identifier type.

C++

```
#define BER_TAG_RELATIVE_OBJECT_IDENTIFIER 13
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.24 BER_TAG_SEQUENCE Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence type.

C++

```
#define BER_TAG_SEQUENCE 16
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.25 BER_TAG_SEQUENCE_OF Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence-of type.

C++

```
#define BER_TAG_SEQUENCE_OF 16
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.26 BER_TAG_SET Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set type.

C++

```
#define BER_TAG_SET 17
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.27 BER_TAG_SET_OF Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set-of type.

C++

```
#define BER_TAG_SET_OF 17
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.28 BER_TAG_T61_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying T61String type.

C++

```
#define BER_TAG_T61_STRING 20
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.29 BER_TAG_TELETEX_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying TeletexString type.

C++

```
#define BER_TAG_TELETEX_STRING 20
```

Module

BerTlv Module (see page 268)

6.1.7.1.3.30 BER_TAG_UNIVERSAL_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UniversalString type.

C++

```
#define BER_TAG_UNIVERSAL_STRING 28
```

ModuleBerTlv Module ([see page 268](#))

6.1.7.1.3.31 BER_TAG_UTC_TIME Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTCTime type.

C++

```
#define BER_TAG_UTC_TIME 23
```

ModuleBerTlv Module ([see page 268](#))

6.1.7.1.3.32 BER_TAG_UTF8_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTF8String type.

C++

```
#define BER_TAG_UTF8_STRING 12
```

ModuleBerTlv Module ([see page 268](#))

6.1.7.1.3.33 BER_TAG_VIDEOTEX_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying VideotexString type.

C++

```
#define BER_TAG_VIDEOTEX_STRING 21
```

ModuleBerTlv Module ([see page 268](#))

6.1.7.1.3.34 BER_TAG_VISIBLE_STRING Macro

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying VisibleString type.

C++

```
#define BER_TAG_VISIBLE_STRING 26
```

ModuleBerTlv Module ([see page 268](#))

6.1.7.1.4 Files

6.1.7.1.4.1 BerTlv.h

Provides functionality for creation, validation, retrieval, update and deletion of primitive and constructed BER-TLV data objects (BerTlv ([see page 268](#))) that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Enumerations

	Name	Description
	BerTagClass_ (see page 297)	Specifies a class of a BER-TLV data object.
	BerTagEncoding_ (see page 297)	Specifies an encoding of a BER-TLV data object.
	BerTagClass (see page 297)	Specifies a class of a BER-TLV data object.
	BerTagEncoding (see page 297)	Specifies an encoding of a BER-TLV data object.

Functions

	Name	Description
	BerTagCreate (see page 272)	Creates BER-TLV tag field.
	BerTagGetClass (see page 273)	Gets the class of the BER-TLV tag field.
	BerTagGetEncoding (see page 273)	Gets the encoding of the BER-TLV data object.
	BerTagGetLength (see page 273)	Gets the length of the BER-TLV tag field.
	BerTagGetNumber (see page 274)	Gets the tag number encoded in the BER-TLV tag field.
	BerTagsValid (see page 274)	Checks whether BER-TLV tag field is valid, i.e. does not violate any of the encoding rules specified in ISO/IEC 7816-4:2005.
	BerTagRead (see page 274)	Reads BER-TLV tag field from memory buffer.
	BerTagWrite (see page 275)	Writes BER-TLV tag field to memory buffer.
	BerTlvClone (see page 275)	Creates a clone of a BER-TLV data object.
	BerTlvCreate (see page 276)	Creates BER-TLV data object.
	BerTlvCreateFromMemory (see page 276)	Creates BER-TLV data object from memory buffer.
	BerTlvGetLength (see page 277)	Gets the BER-TLV data object length field.
	BerTlvGetMinLengthSize (see page 277)	Gets the minimal size of the BER-TLV length field.
	BerTlvGetSize (see page 278)	Gets the size of a BER-TLV data object.
	BerTlvGetSizeMany (see page 278)	Gets the total size of an array of BER-TLV data objects.
	BerTlvGetTag (see page 279)	Gets the BER-TLV data object tag field.
	BerTlvLoadManyFromMemory (see page 279)	Creates an array of BER-TLV data objects from memory buffer.
	BerTlvSaveManyToMemory (see page 280)	Writes an array of BER-TLV data objects to memory buffer.
	BerTlvSaveToMemory (see page 280)	Writes BER-TLV data object to memory buffer.
	BerTlvSetMinLengthSize (see page 281)	Sets the minimal size of the BER-TLV length field.
	ConstructedBerTlvAddDataObject (see page 281)	Creates BER-TLV data object and encapsulates it in a constructed BER-TLV.
	ConstructedBerTlvAddDataObjectCopy (see page 282)	Creates a clone of a BER-TLV data object and encapsulates it in a constructed BER-TLV.
	ConstructedBerTlvAddDataObjectFromMemory (see page 282)	Reads BER-TLV data object from memory buffer and encapsulates it in a constructed BER-TLV.
	ConstructedBerTlvClearDataObjects (see page 283)	Reads BER-TLV data object from memory buffer and encapsulates it in a constructed BER-TLV.
	ConstructedBerTlvFindDataObject (see page 283)	Finds a BER-TLV data object with particular tag in the list of all BER-TLVs that are encapsulated in a constructed BER-TLV.

◆	<code>ConstructedBerTlvFindDataObjectIndex</code> (see page 284)	Finds index of a BER-TLV data object with particular tag in the list of all BER-TLVs that are encapsulated in a constructed BER-TLV.
◆	<code>ConstructedBerTlvGetDataObject</code> (see page 285)	Gets the BER-TLV data object that is encapsulated in a constructed BER-TLV.
◆	<code>ConstructedBerTlvGetDataObjectCapacity</code> (see page 285)	Gets the capacity of all BER-TLV data objects that are encapsulated in a constructed BER-TLV.
◆	<code>ConstructedBerTlvGetDataObjectCount</code> (see page 286)	Gets the number of BER-TLV data objects that are encapsulated in a constructed BER-TLV.
◆	<code>ConstructedBerTlvRemoveDataObject</code> (see page 286)	Removes encapsulated BER-TLV data object from constructed BER-TLV and updates the size of the latter respectively.
◆	<code>ConstructedBerTlvSetDataObjectCapacity</code> (see page 287)	Sets the capacity of all BER-TLV data objects that are encapsulated in a constructed BER-TLV.
◆	<code>PrimitiveBerTlvGetValue</code> (see page 287)	Retrieves the value field of a primitive BER-TLV.
◆	<code>PrimitiveBerTlvGetValueAsByte</code> (see page 287)	Retrieves the value field of a primitive BER-TLV as NByte (see page 116).
◆	<code>PrimitiveBerTlvGetValueAsBoolean</code> (see page 288)	Retrieves the value field of a primitive BER-TLV as NBoolean (see page 116).
◆	<code>PrimitiveBerTlvGetValueAsInt16</code> (see page 288)	Retrieves the value field of a primitive BER-TLV as NInt16 (see page 117).
◆	<code>PrimitiveBerTlvGetValueAsInt32</code> (see page 289)	Retrieves the value field of a primitive BER-TLV as NInt32 (see page 118).
◆	<code>PrimitiveBerTlvGetValueAsInteger</code> (see page 289)	Retrieves the value field of a primitive BER-TLV as integer. Can retrieve from 1 to 4 byte value.
◆	<code>PrimitiveBerTlvGetValueAsSByte</code> (see page 290)	Retrieves the value field of a primitive BER-TLV as NSByte (see page 119).
◆	<code>PrimitiveBerTlvGetValueAsUInt16</code> (see page 290)	Retrieves the value field of a primitive BER-TLV as NUInt16 (see page 120).
◆	<code>PrimitiveBerTlvGetValueAsUInt32</code> (see page 291)	Retrieves the value field of a primitive BER-TLV as NUInt32 (see page 120).
◆	<code>PrimitiveBerTlvGetValuePtr</code> (see page 291)	Gets the pointer to the value field of a primitive BER-TLV.
◆	<code>PrimitiveBerTlvSetLength</code> (see page 292)	Sets the length field of a primitive BER-TLV and updates the size of the BER-TLV respectively.
◆	<code>PrimitiveBerTlvSetValue</code> (see page 292)	Sets the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
◆	<code>PrimitiveBerTlvSetValueAsByte</code> (see page 293)	Sets a NByte (see page 116) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
◆	<code>PrimitiveBerTlvSetValueAsBoolean</code> (see page 293)	Sets a NBoolean (see page 116) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
◆	<code>PrimitiveBerTlvSetValueAsInt16</code> (see page 294)	Sets a NInt16 (see page 117) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
◆	<code>PrimitiveBerTlvSetValueAsInt32</code> (see page 294)	Sets a NInt32 (see page 118) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
◆	<code>PrimitiveBerTlvSetValueAsInteger</code> (see page 295)	Sets an integer to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively. Can set from 1 to 4 byte value.
◆	<code>PrimitiveBerTlvSetValueAsSByte</code> (see page 295)	Sets a NSByte (see page 119) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

	<code>PrimitiveBerTlvSetValueAsUInt16</code> (see page 296)	Sets an <code>NUInt16</code> (see page 120) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.
	<code>PrimitiveBerTlvSetValueAsUInt32</code> (see page 296)	Sets an <code>NUInt32</code> (see page 120) to the value field of a primitive BER-TLV and updates the length field and the size of a BER-TLV respectively.

Macros

Name	Description
<code>BER_TAG_BIT_STRING</code> (see page 297)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying bit string type.
<code>BER_TAG_BMP_STRING</code> (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying BMPString type.
<code>BER_TAG_BOOLEAN</code> (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying boolean type.
<code>BER_TAG_CHARACTER_STRING</code> (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying character string type.
<code>BER_TAG_EMBEDDED_PDV</code> (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying embedded PDV type.
<code>BER_TAG_END_OF_CONTENTS</code> (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying end of contents.
<code>BER_TAG_ENUMERATED</code> (see page 298)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying enumerated type.
<code>BER_TAG_EXTERNAL</code> (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying external type.
<code>BER_TAG_GENERAL_STRING</code> (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GeneralString type.
<code>BER_TAG_GENERALIZED_TIME</code> (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GeneralizedTime type.
<code>BER_TAG_GRAPHIC_STRING</code> (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GraphicString type.
<code>BER_TAG_IA5_STRING</code> (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying IA5String type.
<code>BER_TAG_INSTANCE_OF</code> (see page 299)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying instance-of type.
<code>BER_TAG_INTEGER</code> (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying integer type.
<code>BER_TAG_ISO646_STRING</code> (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying ISO646String type.
<code>BER_TAG_NULL</code> (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying null type.
<code>BER_TAG_NUMERIC_STRING</code> (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying NumericString type.
<code>BER_TAG_OBJECT_DESCRIPTOR</code> (see page 300)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying ObjectDescriptor type.
<code>BER_TAG_OBJECT_IDENTIFIER</code> (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying object identifier type.
<code>BER_TAG_OCTET_STRING</code> (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying octet string type.
<code>BER_TAG_PRINTABLE_STRING</code> (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying PrintableString type.
<code>BER_TAG_REAL</code> (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying real type.

BER_TAG_RELATIVE_OBJECT_IDENTIFIER (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying relative object identifier type.
BER_TAG_SEQUENCE (see page 301)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence type.
BER_TAG_SEQUENCE_OF (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence-of type.
BER_TAG_SET (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set type.
BER_TAG_SET_OF (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set-of type.
BER_TAG_T61_STRING (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying T61String type.
BER_TAG_TELETEX_STRING (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying TeletexString type.
BER_TAG_UNIVERSAL_STRING (see page 302)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UniversalString type.
BER_TAG_UTC_TIME (see page 303)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTCTime type.
BER_TAG_UTF8_STRING (see page 303)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTF8String type.
BER_TAG_VIDEOTEX_STRING (see page 303)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying VideotexString type.
BER_TAG_VISIBLE_STRING (see page 303)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying VisibleString type.

Module

BerTlv Module (see page 268)

6.1.7.2 NSmartCardsBiometry Module

Specifies ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 compliant types and subtypes of biometric data, challenge qualifiers also ISO/IEC 7816-11:2004 and ISO/IEC 7816-6:2004 compliant tag fields of interindustry BER-TLV data objects (BerTlv (see page 268)) used in ISO/IEC 7816-11:2004 Biometric Information Templates (BITs) and Biometric Data Templates (BDTs).

Files

Name	Description
NSmartCardsBiometry.h (see page 318)	Specifies ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 compliant types and subtypes of biometric data, challenge qualifiers also ISO/IEC 7816-11:2004 and ISO/IEC 7816-6:2004 compliant tag fields of interindustry BER-TLV data objects (BerTlv (see page 268)) used in ISO/IEC 7816-11:2004 Biometric Information Templates (BITs) and Biometric Data Templates (BDTs).

Macros

Name	Description
SCARD_BDT_CT_TAG_CHALLENGE (see page 312)	Tag for the challenge data object used in the Challenge Template as specified in ISO/IEC 7816-11:2004.

SCARD_BDT_CT_TAG_CHALLENGE_QUALIFIER (see page 312)	SCARD_BDT_TAG_CHALLENGE_TEMPLATE (see page 313) Tag for the challenge qualifier data object used in the Challenge Template as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_CHALLENGE_DATA (see page 312)	SCARD_TAG_BIOMETRIC_DATA_TEMPLATE (see page 474) Tag for the user prompting challenge data object used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_CHALLENGE_TEMPLATE (see page 313)	Tag for the user prompting challenge template used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA (see page 313)	Tag for the data object containing biometric data with proprietary format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA_OBJECTS (see page 313)	Tag for the template containing biometric data with proprietary format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA (see page 313)	Tag for the data object containing biometric data with standardized format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA_OBJECTS (see page 313)	Tag for the template containing biometric data with standardized format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_DATA (see page 314)	Tag for the biometric matching algorithm parameters data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_TEMPLATE (see page 314)	Tag for the biometric matching algorithm parameters template used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_BIOMETRIC_SUBTYPE (see page 314)	Tag for the biometric subtype data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_BIOMETRIC_TYPE (see page 314)	Tag for the biometric type data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_CREATION_DATE (see page 314)	Tag for the creation date data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

SCARD_BIT_BHT_TAG_CREATOR (see page 315)	Tag for the creator data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_FORMAT_OWNER (see page 315)	Tag for the CBEFF format owner data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_FORMAT_TYPE (see page 315)	Tag for the CBEFF format type data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_INDEX (see page 315)	Tag for the unique reference identifier (index) data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_PATRON_HEADER_VERSION (see page 315)	SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE (see page 317) Tag for the patron header version data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_PRODUCT_IDENTIFIER (see page 316)	Tag for the CBEFF Product Identifier (PID) data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_PROPRIETARY_BHT (see page 316)	Tag for the Biometric Header Template (BHT) that corresponds to the biometric data with proprietary format in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004. Such BHT is encapsulated in the superior layer BHT of the Biometric Information Template (BIT).
SCARD_BIT_BHT_TAG_STANDARD_BHT (see page 316)	Tag for the Biometric Header Template (BHT) that corresponds to the biometric data with standardized format in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004. Such BHT is encapsulated in the superior layer BHT of the Biometric Information Template (BIT).
SCARD_BIT_BHT_TAG_VALIDITY_PERIOD (see page 316)	Tag for the validity period data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_ALGORITHM_REFERENCE (see page 316)	SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE (see page 474) Tag for the algorithm reference data object used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.

SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE (see page 317)	Tag for the CBEFF (ISO/IEC 19785-1:2006) compliant Biometric Header Template (BHT) used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_REFERENCE_DATA_QUALIFIER (see page 317)	Tag for the reference data qualifier data object used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_STANDARD_DATA_OBJECTS (see page 317)	Tag for the template containing standard data objects used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_TAG_ALLOCATION_AUTHORITY_DATA_OBJECTS (see page 317)	Tag for the template containing tag allocation data objects in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.

Structs, Records, Enums

	Name	Description
	SCardBiometricSubtype_ (see page 310)	Specifies the subtype of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardBiometricType_ (see page 311)	Specifies the type of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardChallengeQualifier_ (see page 312)	Specifies challenge qualifiers that may be used in BER-TLV data objects embedded in the ISO/IEC 7816-11:2004 Challenge Template.
	SCardBiometricSubtype (see page 310)	Specifies the subtype of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardBiometricType (see page 311)	Specifies the type of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardChallengeQualifier (see page 312)	Specifies challenge qualifiers that may be used in BER-TLV data objects embedded in the ISO/IEC 7816-11:2004 Challenge Template.

6.1.7.2.1 Structs, Records, Enums

6.1.7.2.1.1 SCardBiometricSubtype Enumeration

Specifies the subtype of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.

C++

```
typedef enum SCardBiometricSubtype_ {
    scbsNone = 0x00,
    scbsRight = 0x01,
    scbsLeft = 0x02,
    scbsThumb = 0x04,
    scbsPointerFinger = 0x08,
    scbsMiddleFinger = 0x0C,
    scbsRingFinger = 0x10,
    scbsLittleFinger = 0x14
} SCardBiometricSubtype;
```

Members

Members	Description
scbsNone = 0x00	No information given.
scbsRight = 0x01	Right.
scbsLeft = 0x02	Left.

scbsThumb = 0x04	Thumb.
scbsPointerFinger = 0x08	Pointer finger.
scbsMiddleFinger = 0x0C	Middle finger.
scbsRingFinger = 0x10	Ring finger.
scbsLittleFinger = 0x14	Little finger.

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.1.2 SCardBiometricType Enumeration

Specifies the type of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.

C++

```
typedef enum SCardBiometricType_ {
    scbtNone = 0x00,
    scbtMultipleBiometrics = 0x01,
    scbtFacialFeatures = 0x02,
    scbtVoice = 0x04,
    scbtFingerprint = 0x08,
    scbtIris = 0x10,
    scbtRetina = 0x20,
    scbtHandGeometry = 0x40,
    scbtSignatureDynamics = 0x80,
    scbtKeystrokeDynamics = 0x0100,
    scbtLipMovement = 0x0200,
    scbtThermalFaceImage = 0x0400,
    scbtThermalHandImage = 0x0800,
    scbtGait = 0x1000,
    scbtBodyOdor = 0x2000,
    scbtDna = 0x4000,
    scbtEarShape = 0x8000,
    scbtFingerGeometry = 0x010000,
    scbtPalmPrint = 0x020000,
    scbtVeinPattern = 0x040000,
    scbtFootPrint = 0x080000
} SCardBiometricType;
```

Members

Members	Description
scbtNone = 0x00	No information given.
scbtMultipleBiometrics = 0x01	Multiple biometrics used.
scbtFacialFeatures = 0x02	Facial features.
scbtVoice = 0x04	Voice.
scbtFingerprint = 0x08	Fingerprint.
scbtIris = 0x10	Iris.
scbtRetina = 0x20	Retina.
scbtHandGeometry = 0x40	Hand geometry.
scbtSignatureDynamics = 0x80	Signature dynamics.
scbtKeystrokeDynamics = 0x0100	Keystroke dynamics.
scbtLipMovement = 0x0200	Lip movement.
scbtThermalFaceImage = 0x0400	Thermal face image.
scbtThermalHandImage = 0x0800	Thermal hand image.
scbtGait = 0x1000	Gait.
scbtBodyOdor = 0x2000	Body odor.
scbtDna = 0x4000	DNA.

scbtEarShape = 0x8000	Ear shape.
scbtFingerGeometry = 0x010000	Finger geometry.
scbtPalmPrint = 0x020000	Palm print.
scbtVeinPattern = 0x040000	Vein pattern.
scbtFootPrint = 0x080000	Foot print.

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.1.3 SCardChallengeQualifier Enumeration

Specifies challenge qualifiers that may be used in BER-TLV data objects embedded in the ISO/IEC 7816-11:2004 Challenge Template.

C++

```
typedef enum SCardChallengeQualifier_ {  
    sccqUnspecified = 0,  
    sccqUtf8Coding = 1,  
} SCardChallengeQualifier;
```

Members

Members	Description
sccqUnspecified = 0	No information given.
sccqUtf8Coding = 1	UTF8 coding.

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2 Macros**6.1.7.2.2.1 SCARD_BDT_CT_TAG_CHALLENGE Macro**

Tag for the challenge data object used in the Challenge Template as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BDT_CT_TAG_CHALLENGE 0x80
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.2 SCARD_BDT_CT_TAG_CHALLENGE_QUALIFIER Macro

SCARD_BDT_TAG_CHALLENGE_TEMPLATE (see page 313)

Tag for the challenge qualifier data object used in the Challenge Template as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BDT_CT_TAG_CHALLENGE_QUALIFIER 0x90
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.3 SCARD_BDT_TAG_CHALLENGE_DATA Macro

SCARD_TAG_BIOMETRIC_DATA_TEMPLATE (see page 474)

Tag for the user prompting challenge data object used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BDT_TAG_CHALLENGE_DATA 0x80
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.4 SCARD_BDT_TAG_CHALLENGE_TEMPLATE Macro

Tag for the user prompting challenge template used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BDT_TAG_CHALLENGE_TEMPLATE 0xA0
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.5 SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA Macro

Tag for the data object containing biometric data with proprietary format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA 0x82
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.6 SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA_OBJECTS Macro

Tag for the template containing biometric data with proprietary format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA_OBJECTS 0xA2
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.7 SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA Macro

Tag for the data object containing biometric data with standardized format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA 0x81
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.8 SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA_OBJECTS Macro

Tag for the template containing biometric data with standardized format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA_OBJECTS 0xA1
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.9 SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_DATA Macro

Tag for the biometric matching algorithm parameters data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_DATA 0x91
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.10

SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_TEMPLATE Macro

Tag for the biometric matching algorithm parameters template used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_TEMPLATE 0xB1
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.11 SCARD_BIT_BHT_TAG_BIOMETRIC_SUBTYPE Macro

Tag for the biometric subtype data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_BIOMETRIC_SUBTYPE 0x82
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.12 SCARD_BIT_BHT_TAG_BIOMETRIC_TYPE Macro

Tag for the biometric type data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_BIOMETRIC_TYPE 0x81
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.13 SCARD_BIT_BHT_TAG_CREATION_DATE Macro

Tag for the creation date data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_CREATION_DATE 0x83
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.14 SCARD_BIT_BHT_TAG_CREATOR Macro

Tag for the creator data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_CREATOR 0x84
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.15 SCARD_BIT_BHT_TAG_FORMAT_OWNER Macro

Tag for the CBEFF format owner data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_FORMAT_OWNER 0x87
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.16 SCARD_BIT_BHT_TAG_FORMAT_TYPE Macro

Tag for the CBEFF format type data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_FORMAT_TYPE 0x88
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.17 SCARD_BIT_BHT_TAG_INDEX Macro

Tag for the unique reference identifier (index) data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_INDEX 0x90
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.18 SCARD_BIT_BHT_TAG_PATRON_HEADER_VERSION Macro

SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE (see page 317)

Tag for the patron header version data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_PATRON_HEADER_VERSION 0x80
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.19 SCARD_BIT_BHT_TAG_PRODUCT_IDENTIFIER Macro

Tag for the CBEFF Product Identifier (PID) data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_PRODUCT_IDENTIFIER 0x86
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.20 SCARD_BIT_BHT_TAG_PROPRIETARY_BHT Macro

Tag for the Biometric Header Template (BHT) that corresponds to the biometric data with proprietary format in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004. Such BHT is encapsulated in the superior layer BHT of the Biometric Information Template (BIT).

C++

```
#define SCARD_BIT_BHT_TAG_PROPRIETARY_BHT 0xA2
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.21 SCARD_BIT_BHT_TAG_STANDARD_BHT Macro

Tag for the Biometric Header Template (BHT) that corresponds to the biometric data with standardized format in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004. Such BHT is encapsulated in the superior layer BHT of the Biometric Information Template (BIT).

C++

```
#define SCARD_BIT_BHT_TAG_STANDARD_BHT 0xA1
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.22 SCARD_BIT_BHT_TAG_VALIDITY_PERIOD Macro

Tag for the validity period data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_BHT_TAG_VALIDITY_PERIOD 0x85
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.23 SCARD_BIT_TAG_ALGORITHM_REFERENCE Macro

SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE (see page 474)

Tag for the algorithm reference data object used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_TAG_ALGORITHM_REFERENCE 0x80
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.24 SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE Macro

Tag for the CBEFF (ISO/IEC 19785-1:2006) compliant Biometric Header Template (BHT) used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE  
SCARD_BIT_TAG_TAG_ALLOCATION_AUTHORITY_DATA_OBJECTS
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.25 SCARD_BIT_TAG_REFERENCE_DATA_QUALIFIER Macro

Tag for the reference data qualifier data object used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_TAG_REFERENCE_DATA_QUALIFIER 0x83
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.26 SCARD_BIT_TAG_STANDARD_DATA_OBJECTS Macro

Tag for the template containing standard data objects used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_TAG_STANDARD_DATA_OBJECTS 0xA0
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.2.27 SCARD_BIT_TAG_TAG_ALLOCATION_AUTHORITY_DATA_OBJECTS Macro

Tag for the template containing tag allocation data objects in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.

C++

```
#define SCARD_BIT_TAG_TAG_ALLOCATION_AUTHORITY_DATA_OBJECTS 0xA1
```

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.2.3 Files

6.1.7.2.3.1 NSmartCardsBiometry.h

Specifies ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 compliant types and subtypes of biometric data, challenge qualifiers also ISO/IEC 7816-11:2004 and ISO/IEC 7816-6:2004 compliant tag fields of interindustry BER-TLV data objects (BerTlv (see page 268)) used in ISO/IEC 7816-11:2004 Biometric Information Templates (BITs) and Biometric Data Templates (BDTs).

Enumerations

	Name	Description
	SCardBiometricSubtype_ (see page 310)	Specifies the subtype of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardBiometricType_ (see page 311)	Specifies the type of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardChallengeQualifier_ (see page 312)	Specifies challenge qualifiers that may be used in BER-TLV data objects embedded in the ISO/IEC 7816-11:2004 Challenge Template.
	SCardBiometricSubtype (see page 310)	Specifies the subtype of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardBiometricType (see page 311)	Specifies the type of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardChallengeQualifier (see page 312)	Specifies challenge qualifiers that may be used in BER-TLV data objects embedded in the ISO/IEC 7816-11:2004 Challenge Template.

Macros

Name	Description
SCARD_BDT_CT_TAG_CHALLENGE (see page 312)	Tag for the challenge data object used in the Challenge Template as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_CT_TAG_CHALLENGE_QUALIFIER (see page 312)	SCARD_BDT_TAG_CHALLENGE_TEMPLATE (see page 313) Tag for the challenge qualifier data object used in the Challenge Template as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_CHALLENGE_DATA (see page 312)	SCARD_TAG_BIOMETRIC_DATA_TEMPLATE (see page 474) Tag for the user prompting challenge data object used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_CHALLENGE_TEMPLATE (see page 313)	Tag for the user prompting challenge template used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA (see page 313)	Tag for the data object containing biometric data with proprietary format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA_OBJECTS (see page 313)	Tag for the template containing biometric data with proprietary format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.

SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA (see page 313)	Tag for the data object containing biometric data with standardized format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA_OBJECTS (see page 313)	Tag for the template containing biometric data with standardized format used in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_DATA (see page 314)	Tag for the biometric matching algorithm parameters data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_TEMPLATE (see page 314)	Tag for the biometric matching algorithm parameters template used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_BIOMETRIC_SUBTYPE (see page 314)	Tag for the biometric subtype data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_BIOMETRIC_TYPE (see page 314)	Tag for the biometric type data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_CREATION_DATE (see page 314)	Tag for the creation date data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_CREATOR (see page 315)	Tag for the creator data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_FORMAT_OWNER (see page 315)	Tag for the CBEFF format owner data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_FORMAT_TYPE (see page 315)	Tag for the CBEFF format type data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_INDEX (see page 315)	Tag for the unique reference identifier (index) data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_PATRON_HEADER_VERSION (see page 315)	SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE (see page 317) Tag for the patron header version data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_BHT_TAG_PRODUCT_IDENTIFIER (see page 316)	Tag for the CBEFF Product Identifier (PID) data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.

SCARD_BIT_BHT_TAG_PROPRIETARY_BHT (see page 316)	Tag for the Biometric Header Template (BHT) that corresponds to the biometric data with proprietary format in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004. Such BHT is encapsulated in the superior layer BHT of the Biometric Information Template (BIT).
SCARD_BIT_BHT_TAG_STANDARD_BHT (see page 316)	Tag for the Biometric Header Template (BHT) that corresponds to the biometric data with standardized format in the Biometric Data Template (BDT) as specified in ISO/IEC 7816-11:2004. Such BHT is encapsulated in the superior layer BHT of the Biometric Information Template (BIT).
SCARD_BIT_BHT_TAG_VALIDITY_PERIOD (see page 316)	Tag for the validity period data object used in the Biometric Header Template (BHT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_ALGORITHM_REFERENCE (see page 316)	SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE (see page 474) Tag for the algorithm reference data object used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE (see page 317)	Tag for the CBEFF (ISO/IEC 19785-1:2006) compliant Biometric Header Template (BHT) used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_REFERENCE_DATA_QUALIFIER (see page 317)	Tag for the reference data qualifier data object used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_STANDARD_DATA_OBJECTS (see page 317)	Tag for the template containing standard data objects used in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.
SCARD_BIT_TAG_TAG_ALLOCATION_AUTHORITY_DATA_OBJECTS (see page 317)	Tag for the template containing tag allocation data objects in the Biometric Information Template (BIT) as specified in ISO/IEC 7816-11:2004.

Module

NSmartCardsBiometry Module (see page 307)

6.1.7.3 NSmartCardsCommands Module

Provides functionality for creation, validation, parsing (e.g., interindustry / proprietary format definition, etc.), transmission and inspection of ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005 compliant APDU commands (C-APDUs) and corresponding APDU responses (R-APDUs).

Files

Name	Description
NSmartCardsCommands.h (see page 409)	Provides functionality for creation, validation, parsing (e.g., interindustry / proprietary format definition, etc.), transmission and inspection of ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005 compliant APDU commands (C-APDUs) and corresponding APDU responses (R-APDUs).

Functions

	Name	Description
≡	ApduClassCreate (see page 371)	Creates class byte (CLA) of the APDU command header according to ISO/IEC 7816-4:2005 specifications. Decision whether first (ISO/IEC 7816-4:2005, table 2) or further (ISO/IEC 7816-4:2005, table 3) interindustry values of CLA should be used is based on the number of the logical channel specified.
≡	ApduClassGetChannelNumber (see page 372)	Retrieves the number of a logical channel used in APDU command with the specified class byte (CLA).
≡	ApduClassGetSecureMessaging (see page 372)	Retrieves the secure messaging mechanism applied to APDU command with specified class byte (CLA).
≡	ApduClassIsInterindustry (see page 372)	Retrieves value indicating whether APDU command with the specified class byte (CLA) is interindustry.
≡	ApduClassIsInterindustryFirst (see page 373)	Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of first interindustry values of CLA.
≡	ApduClassIsInterindustryFurther (see page 373)	Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of further interindustry values of CLA.
≡	ApduClassIsInterindustryReserved (see page 373)	Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of values of CLA reserved for future use by ISO/IEC JTC 1/SC 17.
≡	ApduClassIsLastOrTheOnly (see page 374)	Retrieves value indicating whether APDU command with the specified class byte (CLA) is last or the only command of a chain.
≡	ApduClassIsProprietary (see page 374)	Retrieves value indicating whether APDU command with the specified class byte (CLA) is proprietary.
≡	ApduClassIsValid (see page 374)	Retrieves value indicating whether APDU command with the specified class byte (CLA) is valid, i.e. does not violate any of the CLA encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.
≡	ApduInstructionIsDataBerTlvEncoded (see page 375)	Retrieves value indicating whether data of APDU command with the specified instruction byte (INS) is BER-TLV encoded.
≡	ApduInstructionIsValid (see page 375)	Retrieves value indicating whether APDU command with the specified instruction byte (INS) is valid, i.e. does not violate any of the INS encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.

	<code>ApduStatusGetDataStillAvailableValue</code> (see page 375)	Retrieves the number of data bytes still available from the APDU response status. Applicable only if APDU status is 61XX.
	<code>ApduStatusGetErrorSonvmuTriggeringByTheCardValue</code> (see page 376)	Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 64XX.
	<code>ApduStatusGetErrorWrongLeValue</code> (see page 376)	Retrieves the exact number of available data bytes from the APDU response status. Applicable only if APDU status is 6CXX.
	<code>ApduStatusGetWarningSonvmcCounterValue</code> (see page 376)	Retrieves the value of a counter returned by the card from the APDU response status. Applicable only if APDU status is 63CX.
	<code>ApduStatusGetWarningSonvmuTriggeringByTheCardValue</code> (see page 377)	Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 62XX.
	<code>ApduStatusIsCheckingError</code> (see page 377)	Retrieves value indicating whether APDU response status means checking error.
	<code>ApduStatusIsExecutionError</code> (see page 377)	Retrieves value indicating whether APDU response status means execution error.
	<code>ApduStatusIsInterindustry</code> (see page 378)	Retrieves value indicating whether APDU response status is interindustry.
	<code>ApduStatusIsNormalProcessing</code> (see page 378)	Retrieves value indicating whether APDU response status means normal processing.
	<code>ApduStatusIsProcessAborted</code> (see page 378)	Retrieves value indicating whether APDU response status means process aborted.
	<code>ApduStatusIsProcessComplete</code> (see page 378)	Retrieves value indicating whether APDU response status means process completed.
	<code>ApduStatusIsProprietary</code> (see page 379)	Retrieves value indicating whether APDU response status is proprietary.
	<code>ApduStatusIsStateOfNonVolatileMemoryChanged</code> (see page 379)	Retrieves value indicating whether APDU response status means state of non-volatile memory has changed.
	<code>ApduStatusIsStateOfNonVolatileMemoryUnchanged</code> (see page 379)	Retrieves value indicating whether APDU response status means state of non-volatile memory has changed state of non-volatile memory is unchanged.
	<code>ApduStatusIsValid</code> (see page 380)	Retrieves value indicating whether APDU response status is valid, i.e. does not violate any of the SW1SW2 status bytes encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.
	<code>ApduStatusIsWarningProcessing</code> (see page 380)	Retrieves value indicating whether APDU response status means warning processing.
	<code>SCardTransmitApdu</code> (see page 380)	Transmits APDU command to and retrieves corresponding APDU response from the card terminal.

Macros

Name	Description
APDU_INS_ACTIVATE_FILE (see page 382)	APDU instruction for ACTIVAT E FILE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_APPEND_RECORD (see page 382)	APDU instruction for APPEND RECORD command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_CHANGE_REFERENCE_DATA (see page 382)	APDU instruction for CHANGE REFERE NCE DATA command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_CREATE_FILE (see page 382)	APDU instruction for CREATE FILE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_DEACTIVATE_FILE (see page 383)	APDU instruction for DEACTIV ATE FILE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_DELETE_FILE (see page 383)	APDU instruction for DELETE FILE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_DISABLE_VERIFICATION_REQUIREMENT (see page 383)	APDU instruction for DISABLE VERIFIC ATION REQUIR EMENT command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_ENABLE_VERIFICATION_REQUIREMENT (see page 383)	APDU instruction for ENABLE VERIFICATION REQUIREMENT command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_ENVELOPE (see page 383)	APDU instruction for ENVELOPE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_ENVELOPE_BER_TLV (see page 383)	APDU instruction for ENVELOPE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.

APDU_INS_ERASE_BINARY (🔗 see page 384)	APDU instructon for ERASE BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_ERASE_BINARY_BER_TLV (🔗 see page 384)	APDU instructon for ERASE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_ERASE_RECORD (🔗 see page 384)	APDU instructon for ERASE RECORD (S) command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_EXTERNAL_AUTHENTICATE (see page 384)	APDU instruction for EXTERNAL (/ MUTUAL) AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GENERAL_AUTHENTICATE (see page 384)	APDU instruction for GENERAL AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GENERAL_AUTHENTICATE_BER_TLV (see page 384)	APDU instruction for GENERAL AUTHENTICATE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

APDU_INS_GENERATE_ASYMMETRIC_KEY_PAIR (see page 385)	APDU instruction for GENERATE_ASYMMETRIC_KEY_PAIR command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GET_CHALLENGE (see page 385)	APDU instruction for GET_CHALLENGE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GET_DATA (see page 385)	APDU instruction for GET_DATA command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GET_DATA_BER_TLV (see page 385)	APDU instruction for GET_DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

APDU_INS_GET_RESPONSE (🔗 see page 385)	APDU instructon for GET RESPON SE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_INTERNAL_AUTHENTICATE (🔗 see page 386)	APDU instructon for INTERNA L AUTHEN TICATE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_MANAGE_CHANNEL (🔗 see page 386)	APDU instructon for MANAGE CHANNE L command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_MANAGE_SECURITY_ENVIRONMENT (see page 386)	APDU instructon for MANAGE SECURIT Y ENVIRO NMENT command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_PERFORM_SCQL_OPERATION (see page 386)	APDU instructon for PERFOR M SCQL OPERATI ON command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_PERFORM_SECURITY_OPERATION (see page 386)	APDU instructon for PERFOR M SECURIT Y OPERATI ON command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_PERFORM_TRANSACTION_OPERATION (🔗 see page 386)	APDU instructon for PERFOR M TRANSA CTION OPERATI ON command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_PERFORM_USER_OPERATION (🔗 see page 387)	APDU instructon for PERFOR M USER OPERATI ON command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_PUT_DATA (🔗 see page 387)	APDU instructon for PUT DATA command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_PUT_DATA_BER_TLV (see page 387)	APDU instructon for PUT DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_READ_BINARY (see page 387)	APDU instructon for READ BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_READ_BINARY_BER_TLV (see page 387)	APDU instructon for READ BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.

APDU_INS_READ_RECORD (see page 387)	APDU instructon for READ RECORD (S) command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_READ_RECORD_BER_TLV (see page 388)	APDU instructon for READ RECORD (S) command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_RESET_RETRY_COUNTER (see page 388)	APDU instructon for RESET RETRY COUNT ER command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_SEARCH_BINARY (see page 388)	APDU instructon for SEARCH BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_SEARCH_BINARY_BER_TLV (see page 388)	APDU instructon for SEARCH BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_SEARCH_RECORD (see page 388)	APDU instructon for SEARCH RECORD command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_SELECT (see page 389)	APDU instruction for SELECT command as specified in ISO/IEC 7816-4:2005.
APDU_INS_TERMINATE_CARD_USAGE (see page 389)	APDU instruction for TERMINATE CARD USAGE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_TERMINATE_DF (see page 389)	APDU instruction for TERMINATE DF command as specified in ISO/IEC 7816-4:2005.
APDU_INS_TERMINATE_EF (see page 389)	APDU instruction for TERMINATE EF command as specified in ISO/IEC 7816-4:2005.

APDU_INS_UPDATE_BINARY (🔗 see page 389)	APDU instruction for UPDATE BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_UPDATE_BINARY_BER_TLV (🔗 see page 389)	APDU instruction for UPDATE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_UPDATE_RECORD (🔗 see page 390)	APDU instruction for UPDATE RECORD command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_UPDATE_RECORD_BER_TLV (see page 390)	APDU instructon for UPDATE RECORD command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_VERIFY (see page 390)	APDU instructon for VERIFY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_VERIFY_BER_TLV (see page 390)	APDU instructon for VERIFY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.

APDU_INS_WRITE_BINARY (🔗 see page 390)	APDU instructon for WRITE BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_WRITE_BINARY_BER_TLV (🔗 see page 390)	APDU instructon for WRITE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_WRITE_RECORD (🔗 see page 391)	APDU instructon for WRITE RECORD command as specified in ISO/IEC 7816-4:2 005.

APDU_STATUS_DATA_STILL_AVAILABLE_XX (see page 391)	Mask for APDU status meaning normal processing (SW2 encodes the number of data bytes still available) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CLASS_NOT_SUPPORTED (see page 391)	APDU status meaning checking error class not supported as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_AUTHENTICATION_METHOD_BLOCKED (see page 391)	APDU status meaning checking error command not allowed, authentication method blocked as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_CNA_COMMAND_INCOMPATIBLE_WITH_FILE_STRUCTURE (see page 391)	APDU status meaning checking error command not allowed, command incompatible with file structure as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_COMMAND_NOT_ALLOWED (see page 392)	APDU status meaning checking error command not allowed, command not allowed (no current EF) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_CONDITIONS_OF_USE_NOT_SATISFIED (see page 392)	APDU status meaning checking error command not allowed, conditions of use not satisfied as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_CNA_EXPECTED_SECURE_MESSAGING_DATA_OBJECTS_MISSING (see page 392)	APDU status meaning checking error command not allowed, expected secure messaging data objects missing as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_INCORRECT_SECURE_MESSAGING_DATA_OBJECTS (see page 392)	APDU status meaning checking error command not allowed, incorrect secure messaging data objects as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_NO_INFORMATION (see page 393)	APDU status meaning checking error command not allowed, no information given as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_CNA_REFERENCE_DATA_NOT_USABLE (see page 393)	APDU status meaning checking error command not allowed, reference data not usable as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_SECURITY_STATUS_NOT_SATISFIED (see page 393)	APDU status meaning checking error command not allowed, security status not satisfied as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_XX (see page 393)	Mask for APDU status meaning checking error command not allowed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_FICNS_COMMAND_CHAINING_NOT_SUPPORTED (see page 393)	APDU status meaning checking error functions in CLA not supported, command chaining not supported as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_FICNS_LAST_COMMAND_OF_THE_CHAIN_EXPECTED (see page 394)	APDU status meaning checking error functions in CLA not supported, last command of the chain expected as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_FICNS_LOGICAL_CHANNEL_NOT_SUPPORTED (see page 394)	APDU status meaning checking error functions in CLA not supported, logical channel not supported as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_FICNS_NO_INFORMATION (see page 394)	APDU status meaning checking error functions in CLA not supported, no information given as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_FICNS_SECURE_MESSAGING_NOT_SUPPORTED (see page 394)	APDU status meaning checking error functions in CLA not supported, secure messaging not supported as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_FICNS_XX (see page 394)	Mask for APDU status meaning checking error functions in CLA not supported (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_INSTRUCTION_NOT_SUPPORTED_OR_INVALID (see page 395)	APDU status meaning checking error instruction code not supported or invalid as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_NO_PRECISE_DIAGNOSIS (see page 395)	APDU status meaning checking error no precise diagnosis as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_PROPRIETARY_67XX (see page 395)	Mask for APDU status meaning proprietary checking error 67XX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_PROPRIETARY_6BXX (see page 395)	Mask for APDU status meaning proprietary checking error 6BXX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_PROPRIETARY_6DXX (see page 395)	Mask for APDU status meaning proprietary checking error 6DXX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_PROPRIETARY_6EXX (see page 396)	Mask for APDU status meaning proprietary checking error 6EXX as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_PROPRIETARY_6FXX (see page 396)	Mask for APDU status meaning proprietary checking error 6FXX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SECURITY_XX (see page 396)	Mask for APDU status meaning execution error security-related issues as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SONVMC_MEMORY_FAILURE (see page 396)	APDU status meaning execution error state of non-volatile memory has changed, memory failure as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_SONVMC_NO_INFORMATION (see page 396)	APDU status meaning execution error state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SONVMC_XX (see page 396)	Mask for APDU status meaning execution error state of non-volatile memory has changed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_SONVMU_EXECUTION_ERROR (see page 397)	APDU status meaning execution error state of non-volatile memory is unchanged, execution error as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SONVMU_IMMEDIATE_RESPONSE_REQUIRED_BY_THE_CARD (see page 397)	APDU status meaning execution error state of non-volatile memory is unchanged, immediate response required by the card as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_END (see page 397)	End of values for APDU status meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_START (see page 397)	Start of values for APDU status meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_SONVMU_XX (see page 398)	Mask for APDU status meaning execution error state of non-volatile memory is unchanged (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_DF_NAME_ALREADY_EXISTS (see page 398)	APDU status meaning checking error wrong parameters P1-P2, DF name already exists as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_FILE_ALREADY_EXISTS (see page 398)	APDU status meaning checking error wrong parameters P1-P2, file already exists as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_FILE_OR_APPLICATION_NOT_FOUND (see page 398)	APDU status meaning checking error wrong parameters P1-P2, file or application not found as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_FUNCTION_NOT_SUPPORTED (see page 398)	APDU status meaning checking error wrong parameters P1-P2, function not supported as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_IN_THE_COMMAND_DATA_FIELD (see page 399)	APDU status meaning checking error wrong parameters P1-P2, incorrect parameters in the command data field as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_P1P2 (see page 399)	APDU status meaning checking error wrong parameters P1-P2, incorrect parameters P1-P2 as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_PARAMETERS_P1P2 (see page 399)	APDU status meaning checking error wrong parameters P1-P2, Nc inconsistent with parameters P1-P2 as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_TLV_STRUCTURE (see page 399)	APDU status meaning checking error wrong parameters P1-P2, Nc inconsistent with TLV structure as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_NO_INFORMATION (see page 399)	APDU status meaning checking error wrong parameters P1-P2, no information given as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_NO_QUALIFICATION (see page 400)	APDU status meaning checking error wrong parameters P1-P2, no further indication as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_NOT_ENOUGH_MEMORY_SPACE_IN_THE_FILE (see page 400)	APDU status meaning checking error wrong parameters P1-P2, not enough memory space in the file as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_RECORD_NOT_FOUND (see page 400)	APDU status meaning checking error wrong parameters P1-P2, record not found as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_REFERENCED_DATA_OR_REFERENCE_DATA_NOT_FOUND (see page 400)	APDU status meaning checking error wrong parameters P1-P2, referenced data or reference data not found (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_XX (see page 400)	Mask for APDU status meaning checking error wrong parameters P1-P2 (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WRONG_LE_XX (see page 401)	Mask for APDU status meaning checking error wrong Le field (SW2 encodes the exact number of available data bytes) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WRONG_LENGTH (see page 401)	APDU status meaning checking error wrong length, no further indication as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_NO_QUALIFICATION (see page 401)	APDU status meaning normal processing, no further qualification as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_PROPRIETARY_9XXX (see page 401)	Mask for APDU status meaning proprietary status 9XXX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMC_COUNTER_X (see page 401)	Mask for APDU status meaning warning processing state of non-volatile memory has changed, counter from 0 to 15 encoded by 'X' (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMC_FILE_FILLED_UP_BY_THE_LAST_WRITE (see page 402)	APDU status meaning warning processing state of non-volatile memory has changed, file filled up by the last write as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMC_NO_INFORMATION (see page 402)	APDU status meaning warning processing state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMC_XX (see page 402)	Mask for APDU status meaning warning processing state of non-volatile memory has changed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_END_OF_FILE_OR_RECORD_BEFORE_READING_NE_BYTES (see page 402)	APDU status meaning warning processing state of non-volatile memory is unchanged, end of file or record reached before reading Ne bytes as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_FILE_CONTROL_INFORMATION_INCORRECTY_FORMATTED (see page 402)	APDU status meaning warning processing state of non-volatile memory is unchanged, file control information not formatted according to 5.3.3 as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_NO_INFORMATION (see page 403)	APDU status meaning warning processing state of non-volatile memory is unchanged, no information given as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_NO_INPUT_DATA_AVAILABLE_FROM_A_SENSOR_ON_THE_CARD (see page 403)	APDU status meaning warning processing state of non-volatile memory is unchanged, no input data available from a sensor on the card as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_PART_OF_RETURNED_DATA_MAY_BE_CORRUPTED (see page 403)	APDU status meaning warning processing state of non-volatile memory is unchanged, part of returned data may be corrupted as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_DEACTIVATED (🔗 see page 403)	APDU status meaning warning processing state of non-volatile memory is unchanged, selected file deactivated as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_IN_TERMINATION_STATE (🔗 see page 404)	APDU status meaning warning processing state of non-volatile memory is unchanged, selected file in termination state as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_END (see page 404)	Start of values for APDU status meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_START (see page 404)	End of values for APDU status meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_XX (see page 404)	Mask for APDU status meaning warning processing state of non-volatile memory is unchanged (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
ApduStatusGetSW1 (see page 404)	Retrieves the first byte of the specified APDU status.
ApduStatusGetSW2 (see page 405)	Retrieves the second byte of the specified APDU status.
ApduStatusIsDataStillAvailable (see page 405)	Retrieves a value indicating whether APDU status means normal processing (SW2 encodes the number of data bytes still available).

ApduStatusIsErrorCna (see page 405)	Retrieves a value indicating whether APDU status means checking error command not allowed (further qualification in SW2).
ApduStatusIsErrorFicns (see page 405)	Retrieves a value indicating whether APDU status means checking error functions in CLA not supported (further qualification in SW2).
ApduStatusIsErrorProprietary67XX (see page 405)	Retrieves a value indicating whether APDU status means proprietary error 67XX.

ApduStatusIsErrorProprietary6BXX (↗ see page 406)	Retrieves a value indicating whether APDU status means proprietary error 6BXX.
ApduStatusIsErrorProprietary6DXX (↗ see page 406)	Retrieves a value indicating whether APDU status means proprietary error 6DXX.
ApduStatusIsErrorProprietary6EXX (↗ see page 406)	Retrieves a value indicating whether APDU status means proprietary error 6EXX.
ApduStatusIsErrorProprietary6FXX (↗ see page 406)	Retrieves a value indicating whether APDU status means proprietary error 6FXX.

ApduStatusIsErrorSecurity (see page 406)	Retrieves a value indicating whether APDU status means execution error security-related issues.
ApduStatusIsErrorSonvmc (see page 407)	Retrieves a value indicating whether APDU status means execution error state of non-volatile memory has changed (further qualification in SW2).
ApduStatusIsErrorSonvmu (see page 407)	Retrieves a value indicating whether APDU status means execution error state of non-volatile memory is unchanged (further qualification in SW2).

ApduStatusIsErrorSonvmuTriggeringByTheCard (see page 407)	Retrieves a value indicating whether APDU status means execution error state of non-volatile memory is unchanged, triggering by the card.
ApduStatusIsErrorWP1P2 (see page 407)	Retrieves a value indicating whether APDU status means checking error wrong parameters P1-P2 (further qualification in SW2).

ApduStatusIsErrorWrongLe (see page 407)	Retrieves a value indicating whether APDU status means checking error wrong Le field (SW2 encodes the exact number of available data bytes).
ApduStatusIsProprietary9XXX (see page 408)	Retrieves a value indicating whether APDU status means proprietary status 9XXX.
ApduStatusIsWarningSonvmc (see page 408)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed (further qualification in SW2).

ApduStatusIsWarningSonvmcCounter (see page 408)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed, counter from 0 to 15 (exact meaning depending on the command).
ApduStatusIsWarningSonvmu (see page 408)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged (further qualification in SW2).

ApduStatusIsWarningSonvmuTriggeringByTheCard (🔗 see page 408)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged, triggering by the card.

Structs, Records, Enums

	Name	Description
🔗	ApduClassSecureMessaging_ (🔗 see page 381)	Specifies secure messaging applied to the APDU command-response pairs according to ISO/IEC 7816-4:2005.
	ApduClassSecureMessaging (🔗 see page 381)	Specifies secure messaging applied to the APDU command-response pairs according to ISO/IEC 7816-4:2005.

6.1.7.3.1 Functions

6.1.7.3.1.1 ApduClassCreate Function

Creates class byte (CLA) of the APDU command header according to ISO/IEC 7816-4:2005 specifications. Decision whether first (ISO/IEC 7816-4:2005, table 2) or further (ISO/IEC 7816-4:2005, table 3) interindustry values of CLA should be used is based on the number of the logical channel specified.

C++

```
NResult N_API ApduClassCreate(NBool isLastOrTheOnly, ApduClassSecureMessaging secureMessaging, NInt channelNumber, NByte * pCls);
```

Parameters

Parameters	Description
NBool isLastOrTheOnly	[in] Boolean parameter specifying whether the APDU command is the last or only command of a chain.
ApduClassSecureMessaging secureMessaging	[in] The identifier of a secure messaging mechanism applied to the APDU command.
NInt channelNumber	[in] Number of a logical channel to pass the APDU command through.
NByte * pCls	[out] Pointer to NByte (🔗 see page 116) that receives created class byte (CLA) of the APDU command header.

Returns

If the function succeeds, the return value is N_OK (🔗 see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pCIs</i> is NULL (see page 133).
N_E_ARGUMENT (see page 78)	<i>secureMessaging</i> is not <i>acsmNone</i> and not <i>acsmHeaderNotProcessed</i> . <i>secureMessaging</i> is less than <i>acsmNone</i> or greater than <i>acsmHeaderAuthenticated</i> .
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>channelNumber</i> is less to zero or greater than 19.

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.2 ApduClassGetChannelNumber Function

Retrieves the number of a logical channel used in APDU command with the specified class byte (CLA).

C++

```
NInt N_API ApduClassGetChannelNumber(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.3 ApduClassGetSecureMessaging Function

Retrieves the secure messaging mechanism applied to APDU command with specified class byte (CLA).

C++

```
ApduClassSecureMessaging N_API ApduClassGetSecureMessaging(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.4 ApduClassIsInterindustry Function

Retrieves value indicating whether APDU command with the specified class byte (CLA) is interindustry.

C++

```
NBool N_API ApduClassIsInterindustry(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.5 ApcduClassIsInterindustryFirst Function

Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of first interindustry values of CLA.

C++

```
NBool N_API ApcduClassIsInterindustryFirst(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.6 ApcduClassIsInterindustryFurther Function

Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of further interindustry values of CLA.

C++

```
NBool N_API ApcduClassIsInterindustryFurther(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.7 ApcduClassIsInterindustryReserved Function

Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of values of CLA reserved for future use by ISO/IEC JTC 1/SC 17.

C++

```
NBool N_API ApcduClassIsInterindustryReserved(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.8 ApcduClassIsLastOrTheOnly Function

Retrieves value indicating whether APDU command with the specified class byte (CLA) is last or the only command of a chain.

C++

```
NBool N_API ApcduClassIsLastOrTheOnly(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.9 ApcduClassIsProprietary Function

Retrieves value indicating whether APDU command with the specified class byte (CLA) is proprietary.

C++

```
NBool N_API ApcduClassIsProprietary(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.10 ApcduClassIsValid Function

Retrieves value indicating whether APDU command with the specified class byte (CLA) is valid, i.e. does not violate any of the CLA encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.

C++

```
NBool N_API ApcduClassIsValid(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command class byte (CLA) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.11 AduInstructionIsDataBerTlvEncoded Function

Retrieves value indicating whether data of APDU command with the specified instruction byte (INS) is BER-TLV encoded.

C++

```
NBool N_API AduInstructionIsDataBerTlvEncoded(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command instruction byte (INS) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.12 AduInstructionIsValid Function

Retrieves value indicating whether APDU command with the specified instruction byte (INS) is valid, i.e. does not violate any of the INS encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.

C++

```
NBool N_API AduInstructionIsValid(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] APDU command instruction byte (INS) value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.13 AduStatusGetDataStillAvailableValue Function

Retrieves the number of data bytes still available from the APDU response status. Applicable only if APDU status is 61XX.

C++

```
NInt N_API AduStatusGetDataStillAvailableValue(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.14 ApduStatusGetErrorSonvmuTriggeringByTheCardValue Function

Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 64XX.

C++

```
NInt N_API ApduStatusGetErrorSonvmuTriggeringByTheCardValue(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.15 ApduStatusGetErrorWrongLeValue Function

Retrieves the exact number of available data bytes from the APDU response status. Applicable only if APDU status is 6CXX.

C++

```
NInt N_API ApduStatusGetErrorWrongLeValue(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.16 ApduStatusGetWarningSonvmcCounterValue Function

Retrieves the value of a counter returned by the card from the APDU response status. Applicable only if APDU status is 63CX.

C++

```
NInt N_API ApduStatusGetWarningSonvmcCounterValue(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.1.17 AduStatusGetWarningSonvmuTriggeringByTheCardValue Function

Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 62XX.

C++

```
NInt N_API AduStatusGetWarningSonvmuTriggeringByTheCardValue(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.1.18 AduStatusIsCheckingError Function

Retrieves value indicating whether APDU response status means checking error.

C++

```
NBool N_API AduStatusIsCheckingError(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.1.19 AduStatusIsExecutionError Function

Retrieves value indicating whether APDU response status means execution error.

C++

```
NBool N_API AduStatusIsExecutionError(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.1.20 AduStatusIsInterindustry Function

Retrieves value indicating whether APDU response status is interindustry.

C++

```
NBool N_API AduStatusIsInterindustry(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.21 AduStatusIsNormalProcessing Function

Retrieves value indicating whether APDU response status means normal processing.

C++

```
NBool N_API AduStatusIsNormalProcessing(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.22 AduStatusIsProcessAborted Function

Retrieves value indicating whether APDU response status means process aborted.

C++

```
NBool N_API AduStatusIsProcessAborted(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.23 AduStatusIsProcessComplete Function

Retrieves value indicating whether APDU response status means process completed.

C++

```
NBool N_API ApduStatusIsProcessComplete(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.1.24 ApduStatusIsProprietary Function

Retrieves value indicating whether APDU response status is proprietary.

C++

```
NBool N_API ApduStatusIsProprietary(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.1.25 ApduStatusIsStateOfNonVolatileMemoryChanged Function

Retrieves value indicating whether APDU response status means state of non-volatile memory has changed.

C++

```
NBool N_API ApduStatusIsStateOfNonVolatileMemoryChanged(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.1.26 ApduStatusIsStateOfNonVolatileMemoryUnchanged Function

Retrieves value indicating whether APDU response status means state of non-volatile memory has changed state of non-volatile memory is unchanged.

C++

```
NBool N_API ApduStatusIsStateOfNonVolatileMemoryUnchanged(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.27 AduStatusIsValid Function

Retrieves value indicating whether APDU response status is valid, i.e. does not violate any of the SW1SW2 status bytes encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.

C++

```
NBool N_API AduStatusIsValid(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.28 AduStatusIsWarningProcessing Function

Retrieves value indicating whether APDU response status means warning processing.

C++

```
NBool N_API AduStatusIsWarningProcessing(NUShort value);
```

Parameters

Parameters	Description
NUShort value	[in] APDU response status bytes (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.1.29 SCardTransmitAdu Function

Transmits APDU command to and retrieves corresponding APDU response from the card terminal.

C++

```
NResult N_API SCardTransmitAdu(SCARDHANDLE hCard, NByte cls, NByte instruction, NByte p1, NByte p2, NBool isExtended, const void * pData, NSizeType nc, void * pResponseData, NSizeType ne, NSizeType * pNr, NUShort * pStatus);
```

Parameters

Parameters	Description
SCARDHANDLE hCard	[in] Handle to a smart card.
NByte cls	[in] APDU command class byte (CLA) value.
NByte instruction	[in] APDU command instruction byte (INS) value.
NByte p1	[in] APDU command initial parameter byte (P1) value.
NByte p2	[in] APDU command ulterior parameter byte (P2) value.
NBool isExtended	[in] Boolean parameter specifying whether the APDU command to be send has either extended or short format.
const void * pData	[in] Pointer to memory buffer carrying the data field of APDU command (C-APDU).
NSizeType nc	[in] Length of the APDU command data field.
void * pResponseData	[out] Pointer to memory buffer to carry APDU reponse (R-APDU).
NSizeType ne	[in] Expected length of the APDU response data field.
NSizeType * pNr	[out] Pointer to NSizeType (see page 119) that receives an offset of the APDU response status word (SW1SW2).
NUShort * pStatus	[out] Pointer to NUShort (see page 120) that receives the APDU response status word (SW1SW2).

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>cls</i> or instruction is not valid.
N_E_ARGUMENT_NULL (see page 78)	One of parameter <i>pData</i> , <i>pResponseData</i> , <i>pNr</i> is NULL (see page 133) - and - <i>ne</i> is not equal zero - or - <i>pStatus</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>nc</i> is less than zero or is to big or <i>ne</i> is less than zero or is to big.

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.2 Structs, Records, Enums**6.1.7.3.2.1 ApcduClassSecureMessaging Enumeration**

Specifies secure messaging applied to the APDU command-response pairs according to ISO/IEC 7816-4:2005.

C++

```
typedef enum ApcduClassSecureMessaging_ {
    acsmNone = 0,
    acsmProprietary = 1,
    acsmHeaderNotProcessed = 2,
    acsmHeaderAuthenticated = 3
} ApcduClassSecureMessaging;
```

Members

Members	Description
acsmNone = 0	No indication or no secure messaging applied.
acsmProprietary = 1	Proprietary secure messaging format.
acsmHeaderNotProcessed = 2	Secure messaging according to ISO/IEC 7816-4:2005 section 6. APDU command header not processed according to ISO/IEC 7816-4:2005 section 6.2.3.1.
acsmHeaderAuthenticated = 3	Secure messaging according to ISO/IEC 7816-4:2005 section 6. APDU command header authenticated according to ISO/IEC 7816-4:2005 section 6.2.3.1.

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3 Macros**6.1.7.3.3.1 APDU_INS_ACTIVATE_FILE Macro**

APDU instructon for ACTIVATE FILE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_ACTIVATE_FILE 0x44
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.2 APDU_INS_APPEND_RECORD Macro

APDU instructon for APPEND RECORD command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_APPEND_RECORD 0xE2
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.3 APDU_INS_CHANGE_REFERENCE_DATA Macro

APDU instructon for CHANGE REFERENCE DATA command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_CHANGE_REFERENCE_DATA 0x24
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.4 APDU_INS_CREATE_FILE Macro

APDU instructon for CREATE FILE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_CREATE_FILE 0xE0
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.5 APDU_INS_DEACTIVATE_FILE Macro

APDU instruction for DEACTIVATE FILE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_DEACTIVATE_FILE 0x04
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.6 APDU_INS_DELETE_FILE Macro

APDU instruction for DELETE FILE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_DELETE_FILE 0xE4
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.7 APDU_INS_DISABLE_VERIFICATION_REQUIREMENT Macro

APDU instruction for DISABLE VERIFICATION REQUIREMENT command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_DISABLE_VERIFICATION_REQUIREMENT 0x26
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.8 APDU_INS_ENABLE_VERIFICATION_REQUIREMENT Macro

APDU instruction for ENABLE VERIFICATION REQUIREMENT command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_ENABLE_VERIFICATION_REQUIREMENT 0x28
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.9 APDU_INS_ENVELOPE Macro

APDU instruction for ENVELOPE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_ENVELOPE 0xC2
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.10 APDU_INS_ENVELOPE_BER_TLV Macro

APDU instruction for ENVELOPE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_ENVELOPE_BER_TLV 0xC3
```


Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.11 APDU_INS_ERASE_BINARY Macro

APDU instruction for ERASE BINARY command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_ERASE_BINARY 0x0E
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.12 APDU_INS_ERASE_BINARY_BER_TLV Macro

APDU instruction for ERASE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_ERASE_BINARY_BER_TLV 0x0F
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.13 APDU_INS_ERASE_RECORD Macro

APDU instruction for ERASE RECORD (S) command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_ERASE_RECORD 0x0C
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.14 APDU_INS_EXTERNAL_AUTHENTICATE Macro

APDU instruction for EXTERNAL (/ MUTUAL) AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_EXTERNAL_AUTHENTICATE 0x82
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.15 APDU_INS_GENERAL_AUTHENTICATE Macro

APDU instruction for GENERAL AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_GENERAL_AUTHENTICATE 0x86
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.16 APDU_INS_GENERAL_AUTHENTICATE_BER_TLV Macro

APDU instruction for GENERAL AUTHENTICATE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_GENERAL_AUTHENTICATE_BER_TLV 0x87
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.17 APDU_INS_GENERATE_ASYMMETRIC_KEY_PAIR Macro

APDU instructon for GENERATE ASYMMETRIC KEY PAIR command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_GENERATE_ASYMMETRIC_KEY_PAIR 0x46
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.18 APDU_INS_GET_CHALLENGE Macro

APDU instructon for GET CHALLENGE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_GET_CHALLENGE 0x84
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.19 APDU_INS_GET_DATA Macro

APDU instructon for GET DATA command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_GET_DATA 0xCA
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.20 APDU_INS_GET_DATA_BER_TLV Macro

APDU instructon for GET DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_GET_DATA_BER_TLV 0xCB
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.21 APDU_INS_GET_RESPONSE Macro

APDU instructon for GET RESPONSE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_GET_RESPONSE 0xC0
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.22 APDU_INS_INTERNAL_AUTHENTICATE Macro

APDU instruction for INTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_INTERNAL_AUTHENTICATE 0x88
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.23 APDU_INS_MANAGE_CHANNEL Macro

APDU instruction for MANAGE CHANNEL command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_MANAGE_CHANNEL 0x70
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.24 APDU_INS_MANAGE_SECURITY_ENVIRONMENT Macro

APDU instruction for MANAGE SECURITY ENVIRONMENT command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_MANAGE_SECURITY_ENVIRONMENT 0x22
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.25 APDU_INS_PERFORM_SCQL_OPERATION Macro

APDU instruction for PERFORM SCQL OPERATION command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_PERFORM_SCQL_OPERATION 0x10
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.26 APDU_INS_PERFORM_SECURITY_OPERATION Macro

APDU instruction for PERFORM SECURITY OPERATION command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_PERFORM_SECURITY_OPERATION 0x2A
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.27 APDU_INS_PERFORM_TRANSACTION_OPERATION Macro

APDU instruction for PERFORM TRANSACTION OPERATION command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_PERFORM_TRANSACTION_OPERATION 0x12
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.28 APDU_INS_PERFORM_USER_OPERATION Macro

APDU instructon for PERFORM USER OPERATION command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_PERFORM_USER_OPERATION 0x14
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.29 APDU_INS_PUT_DATA Macro

APDU instructon for PUT DATA command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_PUT_DATA 0xDA
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.30 APDU_INS_PUT_DATA_BER_TLV Macro

APDU instructon for PUT DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_PUT_DATA_BER_TLV 0xDB
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.31 APDU_INS_READ_BINARY Macro

APDU instructon for READ BINARY command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_READ_BINARY 0xB0
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.32 APDU_INS_READ_BINARY_BER_TLV Macro

APDU instructon for READ BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_READ_BINARY_BER_TLV 0xB1
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.33 APDU_INS_READ_RECORD Macro

APDU instructon for READ RECORD (S) command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_READ_RECORD 0xB2
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.34 APDU_INS_READ_RECORD_BER_TLV Macro

APDU instructon for READ RECORD (S) command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_READ_RECORD_BER_TLV 0xB3
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.35 APDU_INS_RESET_RETRY_COUNTER Macro

APDU instructon for RESET RETRY COUNTER command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_RESET_RETRY_COUNTER 0x2C
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.36 APDU_INS_SEARCH_BINARY Macro

APDU instructon for SEARCH BINARY command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_SEARCH_BINARY 0xA0
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.37 APDU_INS_SEARCH_BINARY_BER_TLV Macro

APDU instructon for SEARCH BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_SEARCH_BINARY_BER_TLV 0xA1
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.38 APDU_INS_SEARCH_RECORD Macro

APDU instructon for SEARCH RECORD command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_SEARCH_RECORD 0xA2
```

ModuleNSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.39 APDU_INS_SELECT Macro

APDU instruction for SELECT command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_SELECT 0xA4
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.40 APDU_INS_TERMINATE_CARD_USAGE Macro

APDU instruction for TERMINATE CARD USAGE command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_TERMINATE_CARD_USAGE 0xFE
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.41 APDU_INS_TERMINATE_DF Macro

APDU instruction for TERMINATE DF command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_TERMINATE_DF 0xE6
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.42 APDU_INS_TERMINATE_EF Macro

APDU instruction for TERMINATE EF command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_TERMINATE_EF 0xE8
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.43 APDU_INS_UPDATE_BINARY Macro

APDU instruction for UPDATE BINARY command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_UPDATE_BINARY 0xD6
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.44 APDU_INS_UPDATE_BINARY_BER_TLV Macro

APDU instruction for UPDATE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_UPDATE_BINARY_BER_TLV 0xD7
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.45 APDU_INS_UPDATE_RECORD Macro

APDU instructon for UPDATE RECORD command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_UPDATE_RECORD 0xDC
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.46 APDU_INS_UPDATE_RECORD_BER_TLV Macro

APDU instructon for UPDATE RECORD command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_UPDATE_RECORD_BER_TLV 0xDD
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.47 APDU_INS_VERIFY Macro

APDU instructon for VERIFY command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_VERIFY 0x20
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.48 APDU_INS_VERIFY_BER_TLV Macro

APDU instructon for VERIFY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_VERIFY_BER_TLV 0x21
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.49 APDU_INS_WRITE_BINARY Macro

APDU instructon for WRITE BINARY command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_WRITE_BINARY 0xD0
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.50 APDU_INS_WRITE_BINARY_BER_TLV Macro

APDU instructon for WRITE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_WRITE_BINARY_BER_TLV 0xD1
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.51 APDU_INS_WRITE_RECORD Macro

APDU instruction for WRITE RECORD command as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_INS_WRITE_RECORD 0xD2
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.52 APDU_STATUS_DATA_STILL_AVAILABLE_XX Macro

Mask for APDU status meaning normal processing (SW2 encodes the number of data bytes still available) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_DATA_STILL_AVAILABLE_XX 0x6100
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.53 APDU_STATUS_ERROR_CLASS_NOT_SUPPORTED Macro

APDU status meaning checking error class not supported as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CLASS_NOT_SUPPORTED 0x6E00
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.54 APDU_STATUS_ERROR_CNA_AUTHENTICATION_METHOD_BLOCKED Macro

APDU status meaning checking error command not allowed, authentication method blocked as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_AUTHENTICATION_METHOD_BLOCKED 0x6983
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.55**APDU_STATUS_ERROR_CNA_COMMAND_INCOMPATIBLE_WITH_FILE_STRUCTURE Macro**

APDU status meaning checking error command not allowed, command incompatible with file structure as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_COMMAND_INCOMPATIBLE_WITH_FILE_STRUCTURE 0x6981
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.56 APDU_STATUS_ERROR_CNA_COMMAND_NOT_ALLOWED Macro

APDU status meaning checking error command not allowed, command not allowed (no current EF) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_COMMAND_NOT_ALLOWED 0x6986
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.57 APDU_STATUS_ERROR_CNA_CONDITIONS_OF_USE_NOT_SATISFIED Macro

APDU status meaning checking error command not allowed, conditions of use not satisfied as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_CONDITIONS_OF_USE_NOT_SATISFIED 0x6985
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.58**APDU_STATUS_ERROR_CNA_EXPECTED_SECURE_MESSAGING_DATA_OBJECTS_MISSING Macro**

APDU status meaning checking error command not allowed, expected secure messaging data objects missing as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_EXPECTED_SECURE_MESSAGING_DATA_OBJECTS_MISSING 0x6987
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.59**APDU_STATUS_ERROR_CNA_INCORRECT_SECURE_MESSAGING_DATA_OBJECTS Macro**

APDU status meaning checking error command not allowed, incorrect secure messaging data objects as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_INCORRECT_SECURE_MESSAGING_DATA_OBJECTS 0x6988
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.60 APDU_STATUS_ERROR_CNA_NO_INFORMATION Macro

APDU status meaning checking error command not allowed, no information given as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_NO_INFORMATION 0x6900
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.61 APDU_STATUS_ERROR_CNA_REFERENCE_DATA_NOT_USABLE Macro

APDU status meaning checking error command not allowed, reference data not usable as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_REFERENCE_DATA_NOT_USABLE 0x6984
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.62 APDU_STATUS_ERROR_CNA_SECURITY_STATUS_NOT_SATISFIED Macro

APDU status meaning checking error command not allowed, security status not satisfied as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_SECURITY_STATUS_NOT_SATISFIED 0x6982
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.63 APDU_STATUS_ERROR_CNA_XX Macro

Mask for APDU status meaning checking error command not allowed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_CNA_XX 0x6900
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.64 APDU_STATUS_ERROR_FICNS_COMMAND_CHAINING_NOT_SUPPORTED Macro

APDU status meaning checking error functions in CLA not supported, command chaining not supported as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_FICNS_COMMAND_CHAINING_NOT_SUPPORTED 0x6884
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.65

APDU_STATUS_ERROR_FICNS_LAST_COMMAND_OF_THE_CHAIN_EXPECTED Macro

APDU status meaning checking error functions in CLA not supported, last command of the chain expected as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_FICNS_LAST_COMMAND_OF_THE_CHAIN_EXPECTED 0x6883
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.66 APDU_STATUS_ERROR_FICNS_LOGICAL_CHANNEL_NOT_SUPPORTED Macro

APDU status meaning checking error functions in CLA not supported, logical channel not supported as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_FICNS_LOGICAL_CHANNEL_NOT_SUPPORTED 0x6881
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.67 APDU_STATUS_ERROR_FICNS_NO_INFORMATION Macro

APDU status meaning checking error functions in CLA not supported, no information given as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_FICNS_NO_INFORMATION 0x6800
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.68 APDU_STATUS_ERROR_FICNS_SECURE_MESSAGING_NOT_SUPPORTED Macro

APDU status meaning checking error functions in CLA not supported, secure messaging not supported as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_FICNS_SECURE_MESSAGING_NOT_SUPPORTED 0x6882
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.69 APDU_STATUS_ERROR_FICNS_XX Macro

Mask for APDU status meaning checking error functions in CLA not supported (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_FICNS_XX 0x6800
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.70 APDU_STATUS_ERROR_INSTRUCTION_NOT_SUPPORTED_OR_INVALID Macro

APDU status meaning checking error instruction code not supported or invalid as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_INSTRUCTION_NOT_SUPPORTED_OR_INVALID 0x6D00
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.71 APDU_STATUS_ERROR_NO_PRECISE_DIAGNOSIS Macro

APDU status meaning checking error no precise diagnosis as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_NO_PRECISE_DIAGNOSIS 0x6F00
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.72 APDU_STATUS_ERROR_PROPRIETARY_67XX Macro

Mask for APDU status meaning proprietary checking error 67XX as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_PROPRIETARY_67XX 0x6700
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.73 APDU_STATUS_ERROR_PROPRIETARY_6BXX Macro

Mask for APDU status meaning proprietary checking error 6BXX as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_PROPRIETARY_6BXX 0x6B00
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.74 APDU_STATUS_ERROR_PROPRIETARY_6DXX Macro

Mask for APDU status meaning proprietary checking error 6DXX as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_PROPRIETARY_6DXX 0x6D00
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.75 APDU_STATUS_ERROR_PROPRIETARY_6EXX Macro

Mask for APDU status meaning proprietary checking error 6EXX as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_PROPRIETARY_6EXX 0x6E00
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.76 APDU_STATUS_ERROR_PROPRIETARY_6FXX Macro

Mask for APDU status meaning proprietary checking error 6FXX as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_PROPRIETARY_6FXX 0x6F00
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.77 APDU_STATUS_ERROR_SECURITY_XX Macro

Mask for APDU status meaning execution error security-related issues as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SECURITY_XX 0x6600
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.78 APDU_STATUS_ERROR_SONVMC_MEMORY_FAILURE Macro

APDU status meaning execution error state of non-volatile memory has changed, memory failure as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SONVMC_MEMORY_FAILURE 0x6581
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.79 APDU_STATUS_ERROR_SONVMC_NO_INFORMATION Macro

APDU status meaning execution error state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SONVMC_NO_INFORMATION 0x6500
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.80 APDU_STATUS_ERROR_SONVMC_XX Macro

Mask for APDU status meaning execution error state of non-volatile memory has changed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SONVMC_XX 0x6500
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.81 APDU_STATUS_ERROR_SONVMU_EXECUTION_ERROR Macro

APDU status meaning execution error state of non-volatile memory is unchanged, execution error as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SONVMU_EXECUTION_ERROR 0x6400
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.82**APDU_STATUS_ERROR_SONVMU_IMMEDIATE_RESPONSE_REQUIRED_BY_THE_CARD Macro**

APDU status meaning execution error state of non-volatile memory is unchanged, immediate response required by the card as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SONVMU_IMMEDIATE_RESPONSE_REQUIRED_BY_THE_CARD 0x6401
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.83 APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_END Macro

End of values for APDU status meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_END 0x6480
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.84 APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_START Macro

Start of values for APDU status meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_START 0x6402
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.85 APDU_STATUS_ERROR_SONVMU_XX Macro

Mask for APDU status meaning execution error state of non-volatile memory is unchanged (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_SONVMU_XX 0x6400
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.86 APDU_STATUS_ERROR_WP1P2_DF_NAME_ALREADY_EXISTS Macro

APDU status meaning checking error wrong parameters P1-P2, DF name already exists as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_DF_NAME_ALREADY_EXISTS 0x6A8A
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.87 APDU_STATUS_ERROR_WP1P2_FILE_ALREADY_EXISTS Macro

APDU status meaning checking error wrong parameters P1-P2, file already exists as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_FILE_ALREADY_EXISTS 0x6A89
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.88 APDU_STATUS_ERROR_WP1P2_FILE_OR_APPLICATION_NOT_FOUND Macro

APDU status meaning checking error wrong parameters P1-P2, file or application not found as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_FILE_OR_APPLICATION_NOT_FOUND 0x6A82
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.89 APDU_STATUS_ERROR_WP1P2_FUNCTION_NOT_SUPPORTED Macro

APDU status meaning checking error wrong parameters P1-P2, function not supported as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_FUNCTION_NOT_SUPPORTED 0x6A81
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.90

APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_IN_THE_COMMAND_DATA_FIELD

Macro

APDU status meaning checking error wrong parameters P1-P2, incorrect parameters in the command data field as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_IN_THE_COMMAND_DATA_FIELD 0x6A80
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.91 APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_P1P2 Macro

APDU status meaning checking error wrong parameters P1-P2, incorrect parameters P1-P2 as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_P1P2 0x6A86
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.92

APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_PARAMETERS_P1P2

Macro

APDU status meaning checking error wrong parameters P1-P2, Nc inconsistent with parameters P1-P2 as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_PARAMETERS_P1P2 0x6A87
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.93

APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_TLV_STRUCTURE Macro

APDU status meaning checking error wrong parameters P1-P2, Nc inconsistent with TLV structure as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_TLV_STRUCTURE 0x6A85
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.94 APDU_STATUS_ERROR_WP1P2_NO_INFORMATION Macro

APDU status meaning checking error wrong parameters P1-P2, no information given as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_NO_INFORMATION 0x6A00
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.95 APDU_STATUS_ERROR_WP1P2_NO_QUALIFICATION Macro

APDU status meaning checking error wrong parameters P1-P2, no further indication as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_NO_QUALIFICATION 0x6B00
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.96**APDU_STATUS_ERROR_WP1P2_NOT_ENOUGH_MEMORY_SPACE_IN_THE_FILE Macro**

APDU status meaning checking error wrong parameters P1-P2, not enough memory space in the file as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_NOT_ENOUGH_MEMORY_SPACE_IN_THE_FILE 0x6A84
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.97 APDU_STATUS_ERROR_WP1P2_RECORD_NOT_FOUND Macro

APDU status meaning checking error wrong parameters P1-P2, record not found as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_RECORD_NOT_FOUND 0x6A83
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.98**APDU_STATUS_ERROR_WP1P2_REFERENCED_DATA_OR_REFERENCE_DATA_NOT_FOUND Macro**

APDU status meaning checking error wrong parameters P1-P2, referenced data or reference data not found (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_REFERENCED_DATA_OR_REFERENCE_DATA_NOT_FOUND 0x6A88
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.99 APDU_STATUS_ERROR_WP1P2_XX Macro

Mask for APDU status meaning checking error wrong parameters P1-P2 (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WP1P2_XX 0x6A00
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.100 APDU_STATUS_ERROR_WRONG_LE_XX Macro

Mask for APDU status meaning checking error wrong Le field (SW2 encodes the exact number of available data bytes) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WRONG_LE_XX 0x6C00
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.101 APDU_STATUS_ERROR_WRONG_LENGTH Macro

APDU status meaning checking error wrong length, no further indication as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_ERROR_WRONG_LENGTH 0x6700
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.102 APDU_STATUS_NO_QUALIFICATION Macro

APDU status meaning normal processing, no further qualification as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_NO_QUALIFICATION 0x9000
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.103 APDU_STATUS_PROPRIETARY_9XXX Macro

Mask for APDU status meaning proprietary status 9XXX as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_PROPRIETARY_9XXX 0x9000
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.104 APDU_STATUS_WARNING_SONVMC_COUNTER_X Macro

Mask for APDU status meaning warning processing state of non-volatile memory has changed, counter from 0 to 15 encoded by 'X' (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMC_COUNTER_X 0x63C0
```

Module

NSmartCardsCommands Module ([↗](#) see page 320)

6.1.7.3.3.105

APDU_STATUS_WARNING_SONVMC_FILE_FILLED_UP_BY_THE_LAST_WRITE Macro

APDU status meaning warning processing state of non-volatile memory has changed, file filled up by the last write as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMC_FILE_FILLED_UP_BY_THE_LAST_WRITE 0x6381
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.106 APDU_STATUS_WARNING_SONVMC_NO_INFORMATION Macro

APDU status meaning warning processing state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMC_NO_INFORMATION 0x6300
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.107 APDU_STATUS_WARNING_SONVMC_XX Macro

Mask for APDU status meaning warning processing state of non-volatile memory has changed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMC_XX 0x6300
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.108

APDU_STATUS_WARNING_SONVMU_END_OF_FILE_OR_RECORD_BEFORE_READING_NE_BYTES Macro

APDU status meaning warning processing state of non-volatile memory is unchanged, end of file or record reached before reading Ne bytes as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_END_OF_FILE_OR_RECORD_BEFORE_READING_NE_BYTES 0x6282
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.109

APDU_STATUS_WARNING_SONVMU_FILE_CONTROL_INFORMATION_INCORRECTY_FORMATTED Macro

APDU status meaning warning processing state of non-volatile memory is unchanged, file control information not formatted according to 5.3.3 as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_FILE_CONTROL_INFORMATION_INCORRECTY_FORMATTED 0x6284
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.110 APDU_STATUS_WARNING_SONVMU_NO_INFORMATION Macro

APDU status meaning warning processing state of non-volatile memory is unchanged, no information given as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_NO_INFORMATION 0x6200
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.111**APDU_STATUS_WARNING_SONVMU_NO_INPUT_DATA_AVAILABLE_FROM_A_SENSOR_ON_THE_CARD Macro**

APDU status meaning warning processing state of non-volatile memory is unchanged, no input data available from a sensor on the card as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_NO_INPUT_DATA_AVAILABLE_FROM_A_SENSOR_ON_THE_CARD 0x6286
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.112**APDU_STATUS_WARNING_SONVMU_PART_OF_RETURNED_DATA_MAY_BE_CORRUPTED Macro**

APDU status meaning warning processing state of non-volatile memory is unchanged, part of returned data may be corrupted as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_PART_OF_RETURNED_DATA_MAY_BE_CORRUPTED 0x6281
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.113 APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_DEACTIVATED Macro

APDU status meaning warning processing state of non-volatile memory is unchanged, selected file deactivated as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_DEACTIVATED 0x6283
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.114**APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_IN_TERMINATION_STATE Macro**

APDU status meaning warning processing state of non-volatile memory is unchanged, selected file in termination state as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_IN_TERMINATION_STATE 0x6285
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.115 APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_END Macro

Start of values for APDU status meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_END 0x6280
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.116**APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_START Macro**

End of values for APDU status meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_START 0x6202
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.117 APDU_STATUS_WARNING_SONVMU_XX Macro

Mask for APDU status meaning warning processing state of non-volatile memory is unchanged (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

C++

```
#define APDU_STATUS_WARNING_SONVMU_XX 0x6200
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.118 ApduStatusGetSW1 Macro

Retrieves the first byte of the specified APDU status.

C++

```
#define ApmStatusGetSW1(value) ((NByte)((value) >> 8))
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.119 ApmStatusGetSW2 Macro

Retrieves the second byte of the specified APDU status.

C++

```
#define ApmStatusGetSW2(value) ((NByte)(value))
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.120 ApmStatusIsDataStillAvailable Macro

Retrieves a value indicating whether APDU status means normal processing (SW2 encodes the number of data bytes still available).

C++

```
#define ApmStatusIsDataStillAvailable(value) (((value) & 0xFF00) ==  
APDU_STATUS_DATA_STILL_AVAILABLE_XX)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.121 ApmStatusIsErrorCna Macro

Retrieves a value indicating whether APDU status means checking error command not allowed (further qualification in SW2).

C++

```
#define ApmStatusIsErrorCna(value) (((value) & 0xFF00) == APDU_STATUS_ERROR_CNA_XX)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.122 ApmStatusIsErrorFicns Macro

Retrieves a value indicating whether APDU status means checking error functions in CLA not supported (further qualification in SW2).

C++

```
#define ApmStatusIsErrorFicns(value) (((value) & 0xFF00) == APDU_STATUS_ERROR_FICNS_XX)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.123 ApmStatusIsErrorProprietary67XX Macro

Retrieves a value indicating whether APDU status means proprietary error 67XX.

C++

```
#define ApmStatusIsErrorProprietary67XX(value) ((value) != APDU_STATUS_ERROR_WRONG_LENGTH  
&& ((value) & 0xFF00) == APDU_STATUS_ERROR_PROPRIETARY_67XX)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.124 ApduStatusIsErrorProprietary6BXX Macro

Retrieves a value indicating whether APDU status means proprietary error 6BXX.

C++

```
#define ApduStatusIsErrorProprietary6BXX(value) ((value) !=
APDU_STATUS_ERROR_WP1P2_NO_QUALIFICATION && ((value) & 0xFF00) ==
APDU_STATUS_ERROR_PROPRIETARY_6BXX)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.125 ApduStatusIsErrorProprietary6DXX Macro

Retrieves a value indicating whether APDU status means proprietary error 6DXX.

C++

```
#define ApduStatusIsErrorProprietary6DXX(value) ((value) !=
APDU_STATUS_ERROR_INSTRUCTION_NOT_SUPPORTED_OR_INVALID && ((value) & 0xFF00) ==
APDU_STATUS_ERROR_PROPRIETARY_6DXX)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.126 ApduStatusIsErrorProprietary6EXX Macro

Retrieves a value indicating whether APDU status means proprietary error 6EXX.

C++

```
#define ApduStatusIsErrorProprietary6EXX(value) ((value) !=
APDU_STATUS_ERROR_CLASS_NOT_SUPPORTED && ((value) & 0xFF00) ==
APDU_STATUS_ERROR_PROPRIETARY_6EXX)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.127 ApduStatusIsErrorProprietary6FXX Macro

Retrieves a value indicating whether APDU status means proprietary error 6FXX.

C++

```
#define ApduStatusIsErrorProprietary6FXX(value) ((value) !=
APDU_STATUS_ERROR_NO_PRECISE_DIAGNOSIS && ((value) & 0xFF00) ==
APDU_STATUS_ERROR_PROPRIETARY_6FXX)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.128 ApduStatusIsErrorSecurity Macro

Retrieves a value indicating whether APDU status means execution error security-related issues.

C++

```
#define ApduStatusIsErrorSecurity(value) (((value) & 0xFF00) ==
APDU_STATUS_ERROR_SECURITY_XX)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.129 AduStatusIsErrorSonvmc Macro

Retrieves a value indicating whether APDU status means execution error state of non-volatile memory has changed (further qualification in SW2).

C++

```
#define AduStatusIsErrorSonvmc(value) (((value) & 0xFF00) == APDU_STATUS_ERROR_SONVMC_XX)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.130 AduStatusIsErrorSonvmu Macro

Retrieves a value indicating whether APDU status means execution error state of non-volatile memory is unchanged (further qualification in SW2).

C++

```
#define AduStatusIsErrorSonvmu(value) (((value) & 0xFF00) == APDU_STATUS_ERROR_SONVMU_XX)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.131 AduStatusIsErrorSonvmuTriggeringByTheCard Macro

Retrieves a value indicating whether APDU status means execution error state of non-volatile memory is unchanged, triggering by the card.

C++

```
#define AduStatusIsErrorSonvmuTriggeringByTheCard(value) ((value) >= APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_START && (value) <= APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_END)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.132 AduStatusIsErrorWP1P2 Macro

Retrieves a value indicating whether APDU status means checking error wrong parameters P1-P2 (further qualification in SW2).

C++

```
#define AduStatusIsErrorWP1P2(value) (((value) & 0xFF00) == APDU_STATUS_ERROR_WP1P2_XX)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.3.133 AduStatusIsErrorWrongLe Macro

Retrieves a value indicating whether APDU status means checking error wrong Le field (SW2 encodes the exact number of available data bytes).

C++

```
#define AduStatusIsErrorWrongLe(value) (((value) & 0xFF00) == APDU_STATUS_ERROR_WRONG_LE_XX)
```


Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.134 AduStatusIsProprietary9XXX Macro

Retrieves a value indicating whether APDU status means proprietary status 9XXX.

C++

```
#define AduStatusIsProprietary9XXX(value) ((value) != APDU_STATUS_NO_QUALIFICATION &&
((value) & 0xF000) == APDU_STATUS_PROPRIETARY_9XXX)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.135 AduStatusIsWarningSonvmc Macro

Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed (further qualification in SW2).

C++

```
#define AduStatusIsWarningSonvmc(value) (((value) & 0xFF00) ==
APDU_STATUS_WARNING_SONVMC_XX)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.136 AduStatusIsWarningSonvmcCounter Macro

Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed, counter from 0 to 15 (exact meaning depending on the command).

C++

```
#define AduStatusIsWarningSonvmcCounter(value) (((value) & 0xFFF0) ==
APDU_STATUS_WARNING_SONVMC_COUNTER_X)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.137 AduStatusIsWarningSonvmu Macro

Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged (further qualification in SW2).

C++

```
#define AduStatusIsWarningSonvmu(value) (((value) & 0xFF00) ==
APDU_STATUS_WARNING_SONVMU_XX)
```

Module

NSmartCardsCommands Module (see page 320)

6.1.7.3.3.138 AduStatusIsWarningSonvmuTriggeringByTheCard Macro

Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged, triggering by the card.

C++

```
#define AduStatusIsWarningSonvmuTriggeringByTheCard(value) ((value) >=
```

```
APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_START && (value) <=
APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_END)
```

Module

NSmartCardsCommands Module ([see page 320](#))

6.1.7.3.4 Files

6.1.7.3.4.1 NSmartCardsCommands.h

Provides functionality for creation, validation, parsing (e.g., interindustry / proprietary format definition, etc.), transmission and inspection of ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005 compliant APDU commands (C-APDUs) and corresponding APDU responses (R-APDUs).

Enumerations

	Name	Description
	ApduClassSecureMessaging_ (see page 381)	Specifies secure messaging applied to the APDU command-response pairs according to ISO/IEC 7816-4:2005.
	ApduClassSecureMessaging (see page 381)	Specifies secure messaging applied to the APDU command-response pairs according to ISO/IEC 7816-4:2005.

Functions

	Name	Description
	ApduClassCreate (see page 371)	Creates class byte (CLA) of the APDU command header according to ISO/IEC 7816-4:2005 specifications. Decision whether first (ISO/IEC 7816-4:2005, table 2) or further (ISO/IEC 7816-4:2005, table 3) interindustry values of CLA should be used is based on the number of the logical channel specified.
	ApduClassGetChannelNumber (see page 372)	Retrieves the number of a logical channel used in APDU command with the specified class byte (CLA).
	ApduClassGetSecureMessaging (see page 372)	Retrieves the secure messaging mechanism applied to APDU command with specified class byte (CLA).
	ApduClassIsInterindustry (see page 372)	Retrieves value indicating whether APDU command with the specified class byte (CLA) is interindustry.
	ApduClassIsInterindustryFirst (see page 373)	Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of first interindustry values of CLA.
	ApduClassIsInterindustryFurther (see page 373)	Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of further interindustry values of CLA.
	ApduClassIsInterindustryReserved (see page 373)	Retrieves value indicating whether APDU command with the specified class byte (CLA) belongs to the range of values of CLA reserved for future use by ISO/IEC JTC 1/SC 17.
	ApduClassIsLastOrTheOnly (see page 374)	Retrieves value indicating whether APDU command with the specified class byte (CLA) is last or the only command of a chain.

ApduClassIsProprietary (see page 374)	Retrieves value indicating whether APDU command with the specified class byte (CLA) is proprietary.
ApduClassIsValid (see page 374)	Retrieves value indicating whether APDU command with the specified class byte (CLA) is valid, i.e. does not violate any of the CLA encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.
ApduInstructionIsDataBerTlvEncoded (see page 375)	Retrieves value indicating whether data of APDU command with the specified instruction byte (INS) is BER-TLV encoded.
ApduInstructionIsValid (see page 375)	Retrieves value indicating whether APDU command with the specified instruction byte (INS) is valid, i.e. does not violate any of the INS encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.
ApduStatusGetDataStillAvailableValue (see page 375)	Retrieves the number of data bytes still available from the APDU response status. Applicable only if APDU status is 61XX.
ApduStatusGetErrorSonvmuTriggeringByTheCardValue (see page 376)	Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 64XX.
ApduStatusGetErrorWrongLeValue (see page 376)	Retrieves the exact number of available data bytes from the APDU response status. Applicable only if APDU status is 6CXX.
ApduStatusGetWarningSonvmcCounterValue (see page 376)	Retrieves the value of a counter returned by the card from the APDU response status. Applicable only if APDU status is 63CX.
ApduStatusGetWarningSonvmuTriggeringByTheCardValue (see page 377)	Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 62XX.
ApduStatusIsCheckingError (see page 377)	Retrieves value indicating whether APDU response status means checking error.
ApduStatusIsExecutionError (see page 377)	Retrieves value indicating whether APDU response status means execution error.
ApduStatusIsInterindustry (see page 378)	Retrieves value indicating whether APDU response status is interindustry.
ApduStatusIsNormalProcessing (see page 378)	Retrieves value indicating whether APDU response status means normal processing.
ApduStatusIsProcessAborted (see page 378)	Retrieves value indicating whether APDU response status means process aborted.
ApduStatusIsProcessComplete (see page 378)	Retrieves value indicating whether APDU response status means process completed.
ApduStatusIsProprietary (see page 379)	Retrieves value indicating whether APDU response status is proprietary.
ApduStatusIsStateOfNonVolatileMemoryChanged (see page 379)	Retrieves value indicating whether APDU response status means state of non-volatile memory has changed.
ApduStatusIsStateOfNonVolatileMemoryUnchanged (see page 379)	Retrieves value indicating whether APDU response status means state of non-volatile memory has changed state of non-volatile memory is unchanged.

	ApduStatusIsValid (see page 380)	Retrieves value indicating whether APDU response status is valid, i.e. does not violate any of the SW1SW2 status bytes encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.
	ApduStatusIsWarningProcessing (see page 380)	Retrieves value indicating whether APDU response status means warning processing.
	SCardTransmitApdu (see page 380)	Transmits APDU command to and retrieves corresponding APDU response from the card terminal.

Macros

Name	Description
APDU_INS_ACTIVATE_FILE (see page 382)	APDU instruction for ACTIVATE FILE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_APPEND_RECORD (see page 382)	APDU instruction for APPEND RECORD command as specified in ISO/IEC 7816-4:2005.
APDU_INS_CHANGE_REFERENCE_DATA (see page 382)	APDU instruction for CHANGE REFERENCE DATA command as specified in ISO/IEC 7816-4:2005.

APDU_INS_CREATE_FILE (see page 382)	APDU instruction for CREATE FILE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_DEACTIVATE_FILE (see page 383)	APDU instruction for DEACTIV ATE FILE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_DELETE_FILE (see page 383)	APDU instruction for DELETE FILE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_DISABLE_VERIFICATION_REQUIREMENT (see page 383)	APDU instruction for DISABLE VERIFIC ATION REQUIR EMENT command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_ENABLE_VERIFICATION_REQUIREMENT (see page 383)	APDU instruction for ENABLE VERIFICATION REQUIREMENT command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_ENVELOPE (see page 383)	APDU instruction for ENVELOPE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_ENVELOPE_BER_TLV (see page 383)	APDU instruction for ENVELOPE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.

APDU_INS_ERASE_BINARY (see page 384)	APDU instruction for ERASE BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_ERASE_BINARY_BER_TLV (see page 384)	APDU instruction for ERASE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_ERASE_RECORD (see page 384)	APDU instruction for ERASE RECORD (S) command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_EXTERNAL_AUTHENTICATE (see page 384)	APDU instruction for EXTERNAL (/ MUTUAL) AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GENERAL_AUTHENTICATE (see page 384)	APDU instruction for GENERAL AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GENERAL_AUTHENTICATE_BER_TLV (see page 384)	APDU instruction for GENERAL AUTHENTICATE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

APDU_INS_GENERATE_ASYMMETRIC_KEY_PAIR (see page 385)	APDU instruction for GENERATE_ASYMMETRIC_KEY_PAIR command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GET_CHALLENGE (see page 385)	APDU instruction for GET_CHALLENGE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GET_DATA (see page 385)	APDU instruction for GET_DATA command as specified in ISO/IEC 7816-4:2005.
APDU_INS_GET_DATA_BER_TLV (see page 385)	APDU instruction for GET_DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

APDU_INS_GET_RESPONSE (🔗 see page 385)	APDU instructon for GET RESPON SE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_INTERNAL_AUTHENTICATE (🔗 see page 386)	APDU instructon for INTERNA L AUTHEN TICATE command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_MANAGE_CHANNEL (🔗 see page 386)	APDU instructon for MANAGE CHANNE L command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_MANAGE_SECURITY_ENVIRONMENT (see page 386)	APDU instruction for MANAGE SECURITY ENVIRONMENT command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_PERFORM_SCQL_OPERATION (see page 386)	APDU instruction for PERFORM SCQL OPERATION command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_PERFORM_SECURITY_OPERATION (see page 386)	APDU instruction for PERFORM SECURITY OPERATION command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_PERFORM_TRANSACTION_OPERATION (↗ see page 386)	APDU instructon for PERFOR M TRANSA CTION OPERATI ON command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_PERFORM_USER_OPERATION (↗ see page 387)	APDU instructon for PERFOR M USER OPERATI ON command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_PUT_DATA (↗ see page 387)	APDU instructon for PUT DATA command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_PUT_DATA_BER_TLV (see page 387)	APDU instructon for PUT DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_READ_BINARY (see page 387)	APDU instructon for READ BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_READ_BINARY_BER_TLV (see page 387)	APDU instructon for READ BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.

APDU_INS_READ_RECORD (see page 387)	APDU instructon for READ RECORD (S) command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_READ_RECORD_BER_TLV (see page 388)	APDU instructon for READ RECORD (S) command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_RESET_RETRY_COUNTER (see page 388)	APDU instructon for RESET RETRY COUNT ER command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_SEARCH_BINARY (see page 388)	APDU instructon for SEARCH BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_SEARCH_BINARY_BER_TLV (see page 388)	APDU instructon for SEARCH BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_SEARCH_RECORD (see page 388)	APDU instructon for SEARCH RECORD command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_SELECT (see page 389)	APDU instruction for SELECT command as specified in ISO/IEC 7816-4:2005.
APDU_INS_TERMINATE_CARD_USAGE (see page 389)	APDU instruction for TERMINATE CARD USAGE command as specified in ISO/IEC 7816-4:2005.
APDU_INS_TERMINATE_DF (see page 389)	APDU instruction for TERMINATE DF command as specified in ISO/IEC 7816-4:2005.
APDU_INS_TERMINATE_EF (see page 389)	APDU instruction for TERMINATE EF command as specified in ISO/IEC 7816-4:2005.

APDU_INS_UPDATE_BINARY (🔗 see page 389)	APDU instruction for UPDATE BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_UPDATE_BINARY_BER_TLV (🔗 see page 389)	APDU instruction for UPDATE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_UPDATE_RECORD (🔗 see page 390)	APDU instruction for UPDATE RECORD command as specified in ISO/IEC 7816-4:2 005.

APDU_INS_UPDATE_RECORD_BER_TLV (see page 390)	APDU instruction for UPDATE RECORD command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_VERIFY (see page 390)	APDU instruction for VERIFY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_VERIFY_BER_TLV (see page 390)	APDU instruction for VERIFY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.

APDU_INS_WRITE_BINARY (see page 390)	APDU instruction for WRITE BINARY command as specified in ISO/IEC 7816-4:2 005.
APDU_INS_WRITE_BINARY_BER_TLV (see page 390)	APDU instruction for WRITE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2 005.
APDU_INS_WRITE_RECORD (see page 391)	APDU instruction for WRITE RECORD command as specified in ISO/IEC 7816-4:2 005.

APDU_STATUS_DATA_STILL_AVAILABLE_XX (see page 391)	Mask for APDU status meaning normal processing (SW2 encodes the number of data bytes still available) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CLASS_NOT_SUPPORTED (see page 391)	APDU status meaning checking error class not supported as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_AUTHENTICATION_METHOD_BLOCKED (see page 391)	APDU status meaning checking error command not allowed, authentication method blocked as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_CNA_COMMAND_INCOMPATIBLE_WITH_FILE_STRUCTURE (see page 391)	APDU status meaning checking error command not allowed, command incompatible with file structure as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_COMMAND_NOT_ALLOWED (see page 392)	APDU status meaning checking error command not allowed, command not allowed (no current EF) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_CONDITIONS_OF_USE_NOT_SATISFIED (see page 392)	APDU status meaning checking error command not allowed, conditions of use not satisfied as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_CNA_EXPECTED_SECURE_MESSAGING_DATA_OBJECTS_MISSING (see page 392)	APDU status meaning checking error command not allowed, expected secure messaging data objects missing as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_INCORRECT_SECURE_MESSAGING_DATA_OBJECTS (see page 392)	APDU status meaning checking error command not allowed, incorrect secure messaging data objects as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_NO_INFORMATION (see page 393)	APDU status meaning checking error command not allowed, no information given as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_CNA_REFERENCE_DATA_NOT_USABLE (see page 393)	APDU status meaning checking error command not allowed, reference data not usable as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_SECURITY_STATUS_NOT_SATISFIED (see page 393)	APDU status meaning checking error command not allowed, security status not satisfied as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_CNA_XX (see page 393)	Mask for APDU status meaning checking error command not allowed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_FICNS_COMMAND_CHAINING_NOT_SUPPORTED (see page 393)	APDU status meaning checking error functions in CLA not supported, command chaining not supported as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_FICNS_LAST_COMMAND_OF_THE_CHAIN_EXPECTED (see page 394)	APDU status meaning checking error functions in CLA not supported, last command of the chain expected as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_FICNS_LOGICAL_CHANNEL_NOT_SUPPORTED (see page 394)	APDU status meaning checking error functions in CLA not supported, logical channel not supported as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_FICNS_NO_INFORMATION (see page 394)	APDU status meaning checking error functions in CLA not supported, no information given as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_FICNS_SECURE_MESSAGING_NOT_SUPPORTED (see page 394)	APDU status meaning checking error functions in CLA not supported, secure messaging not supported as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_FICNS_XX (see page 394)	Mask for APDU status meaning checking error functions in CLA not supported (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_INSTRUCTION_NOT_SUPPORTED_OR_INVALID (see page 395)	APDU status meaning checking error instruction code not supported or invalid as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_NO_PRECISE_DIAGNOSIS (see page 395)	APDU status meaning checking error no precise diagnosis as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_PROPRIETARY_67XX (see page 395)	Mask for APDU status meaning proprietary checking error 67XX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_PROPRIETARY_6BXX (see page 395)	Mask for APDU status meaning proprietary checking error 6BXX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_PROPRIETARY_6DXX (see page 395)	Mask for APDU status meaning proprietary checking error 6DXX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_PROPRIETARY_6EXX (see page 396)	Mask for APDU status meaning proprietary checking error 6EXX as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_PROPRIETARY_6FXX (see page 396)	Mask for APDU status meaning proprietary checking error 6FXX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SECURITY_XX (see page 396)	Mask for APDU status meaning execution error security-related issues as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SONVMC_MEMORY_FAILURE (see page 396)	APDU status meaning execution error state of non-volatile memory has changed, memory failure as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_SONVMC_NO_INFORMATION (see page 396)	APDU status meaning execution error state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SONVMC_XX (see page 396)	Mask for APDU status meaning execution error state of non-volatile memory has changed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_SONVMU_EXECUTION_ERROR (see page 397)	APDU status meaning execution error state of non-volatile memory is unchanged, execution error as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SONVMU_IMMEDIATE_RESPONSE_REQUIRED_BY_THE_CARD (see page 397)	APDU status meaning execution error state of non-volatile memory is unchanged, immediate response required by the card as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_END (see page 397)	End of values for APDU status meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_START (see page 397)	Start of values for APDU status meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_SONVMU_XX (see page 398)	Mask for APDU status meaning execution error state of non-volatile memory is unchanged (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_DF_NAME_ALREADY_EXISTS (see page 398)	APDU status meaning checking error wrong parameters P1-P2, DF name already exists as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_FILE_ALREADY_EXISTS (see page 398)	APDU status meaning checking error wrong parameters P1-P2, file already exists as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_FILE_OR_APPLICATION_NOT_FOUND (see page 398)	APDU status meaning checking error wrong parameters P1-P2, file or application not found as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_FUNCTION_NOT_SUPPORTED (see page 398)	APDU status meaning checking error wrong parameters P1-P2, function not supported as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_IN_THE_COMMAND_DATA_FIELD (see page 399)	APDU status meaning checking error wrong parameters P1-P2, incorrect parameters in the command data field as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_P1P2 (see page 399)	APDU status meaning checking error wrong parameters P1-P2, incorrect parameters P1-P2 as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_PARAMETERS_P1P2 (see page 399)	APDU status meaning checking error wrong parameters P1-P2, Nc inconsistent with parameters P1-P2 as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_TLV_STRUCTURE (see page 399)	APDU status meaning checking error wrong parameters P1-P2, Nc inconsistent with TLV structure as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_NO_INFORMATION (see page 399)	APDU status meaning checking error wrong parameters P1-P2, no information given as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_NO_QUALIFICATION (see page 400)	APDU status meaning checking error wrong parameters P1-P2, no further indication as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_NOT_ENOUGH_MEMORY_SPACE_IN_THE_FILE (see page 400)	APDU status meaning checking error wrong parameters P1-P2, not enough memory space in the file as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_RECORD_NOT_FOUND (see page 400)	APDU status meaning checking error wrong parameters P1-P2, record not found as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WP1P2_REFERENCED_DATA_OR_REFERENCE_DATA_NOT_FOUND (see page 400)	APDU status meaning checking error wrong parameters P1-P2, referenced data or reference data not found (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_ERROR_WP1P2_XX (see page 400)	Mask for APDU status meaning checking error wrong parameters P1-P2 (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WRONG_LE_XX (see page 401)	Mask for APDU status meaning checking error wrong Le field (SW2 encodes the exact number of available data bytes) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_ERROR_WRONG_LENGTH (see page 401)	APDU status meaning checking error wrong length, no further indication as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_NO_QUALIFICATION (see page 401)	APDU status meaning normal processing, no further qualification as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_PROPRIETARY_9XXX (see page 401)	Mask for APDU status meaning proprietary status 9XXX as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMC_COUNTER_X (see page 401)	Mask for APDU status meaning warning processing state of non-volatile memory has changed, counter from 0 to 15 encoded by 'X' (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMC_FILE_FILLED_UP_BY_THE_LAST_WRITE (see page 402)	APDU status meaning warning processing state of non-volatile memory has changed, file filled up by the last write as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMC_NO_INFORMATION (see page 402)	APDU status meaning warning processing state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMC_XX (see page 402)	Mask for APDU status meaning warning processing state of non-volatile memory has changed (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_END_OF_FILE_OR_RECORD_BEFORE_READING_NE_BYTES (see page 402)	APDU status meaning warning processing state of non-volatile memory is unchanged, end of file or record reached before reading Ne bytes as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_FILE_CONTROL_INFORMATION_INCORRECTY_FORMATTED (see page 402)	APDU status meaning warning processing state of non-volatile memory is unchanged, file control information not formatted according to 5.3.3 as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_NO_INFORMATION (see page 403)	APDU status meaning warning processing state of non-volatile memory is unchanged, no information given as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_NO_INPUT_DATA_AVAILABLE_FROM_A_SENSOR_ON_THE_CARD (see page 403)	APDU status meaning warning processing state of non-volatile memory is unchanged, no input data available from a sensor on the card as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_PART_OF_RETURNED_DATA_MAY_BE_CORRUPTED (see page 403)	APDU status meaning warning processing state of non-volatile memory is unchanged, part of returned data may be corrupted as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_DEACTIVATED (🔗 see page 403)	APDU status meaning warning processing state of non-volatile memory is unchanged, selected file deactivated as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_IN_TERMINATION_STATE (🔗 see page 404)	APDU status meaning warning processing state of non-volatile memory is unchanged, selected file in termination state as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_END (↗ see page 404)	Start of values for APDU status meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.
APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_START (↗ see page 404)	End of values for APDU status meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

APDU_STATUS_WARNING_SONVMU_XX (see page 404)	Mask for APDU status meaning warning processing state of non-volatile memory is unchanged (further qualification in SW2) as specified in ISO/IEC 7816-4:2005.
ApduStatusGetSW1 (see page 404)	Retrieves the first byte of the specified APDU status.
ApduStatusGetSW2 (see page 405)	Retrieves the second byte of the specified APDU status.
ApduStatusIsDataStillAvailable (see page 405)	Retrieves a value indicating whether APDU status means normal processing (SW2 encodes the number of data bytes still available).

ApduStatusIsErrorCna ( see page 405)	Retrieves a value indicating whether APDU status means checking error command not allowed (further qualification in SW2).
ApduStatusIsErrorFicns ( see page 405)	Retrieves a value indicating whether APDU status means checking error functions in CLA not supported (further qualification in SW2).
ApduStatusIsErrorProprietary67XX ( see page 405)	Retrieves a value indicating whether APDU status means proprietary error 67XX.

ApduStatusIsErrorProprietary6BXX (↗ see page 406)	Retrieves a value indicating whether APDU status means proprietary error 6BXX.
ApduStatusIsErrorProprietary6DXX (↗ see page 406)	Retrieves a value indicating whether APDU status means proprietary error 6DXX.
ApduStatusIsErrorProprietary6EXX (↗ see page 406)	Retrieves a value indicating whether APDU status means proprietary error 6EXX.
ApduStatusIsErrorProprietary6FXX (↗ see page 406)	Retrieves a value indicating whether APDU status means proprietary error 6FXX.

ApduStatusIsErrorSecurity (see page 406)	Retrieves a value indicating whether APDU status means execution error security-related issues.
ApduStatusIsErrorSonvmc (see page 407)	Retrieves a value indicating whether APDU status means execution error state of non-volatile memory has changed (further qualification in SW2).
ApduStatusIsErrorSonvmu (see page 407)	Retrieves a value indicating whether APDU status means execution error state of non-volatile memory is unchanged (further qualification in SW2).

ApduStatusIsErrorSonvmuTriggeringByTheCard (see page 407)	Retrieves a value indicating whether APDU status means execution error state of non-volatile memory is unchanged, triggering by the card.
ApduStatusIsErrorWP1P2 (see page 407)	Retrieves a value indicating whether APDU status means checking error wrong parameters P1-P2 (further qualification in SW2).

ApduStatusIsErrorWrongLe (see page 407)	Retrieves a value indicating whether APDU status means checking error wrong Le field (SW2 encodes the exact number of available data bytes).
ApduStatusIsProprietary9XXX (see page 408)	Retrieves a value indicating whether APDU status means proprietary status 9XXX.
ApduStatusIsWarningSonvmc (see page 408)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed (further qualification in SW2).

ApduStatusIsWarningSonvmcCounter (see page 408)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed, counter from 0 to 15 (exact meaning depending on the command).
ApduStatusIsWarningSonvmu (see page 408)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged (further qualification in SW2).

ApduStatusIsWarningSonvmuTriggeringByTheCard (see page 408)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged, triggering by the card.

Module

NSmartCardsCommands Module (see page 320)

6.1.7.4 NSmartCardsDataElements Module

Specifies tags of ISO/IEC 7816-6:2004 interindustry data elements used for interchange.

Files

Name	Description
NSmartCardsDataElements.h (see page 493)	Specifies tags of ISO/IEC 7816-6:2004 interindustry data elements used for interchange.

Macros

Name	Description
SCARD_TAG_ACCOUNT_TYPE (see page 471)	Tag for data object that indicates the type of account selected by the cardholder for the transaction (see EMV for coding) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_ADDRESS (see page 471)	Tag for data object containing address of an individual as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_ANSWER_TO_RESET (see page 471)	Tag for data object that indicates operating characteristics of the card as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_APPLICATION_EFFECTIVE_DATE (see page 471)	Tag for data object containing date from which the application can be used, under the responsibility of the application provider as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_EXPIRATION_DATE (see page 472)	Tag for data object containing date after which an application expires, under the responsibility of the application provider as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_FAMILY_IDENTIFIER (see page 472)	Tag for data object containing designation of several application areas to enable global interoperability (see ISO/IEC 14443) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_APPLICATION_IDENTIFIER (see page 472)	Tag for data object containing data element identifying an application in the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_IMAGE (see page 472)	Tag for data object containing icon or logo associated with an application as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_IMAGE_TEMPLATE (see page 472)	Tag of template nesting at least an application image as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_LABEL (see page 473)	Tag for data element for use at the man-machine interface as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_RELATED_DATA (see page 473)	Tag for template nesting parameters of an application as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_TEMPLATE (see page 473)	Tag for template identifying an application in the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_TRACK_1 (see page 473)	Tag for data object containing structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_APPLICATION_TRACK_2 (see page 473)	Tag for data object containing structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_TRACK_3 (see page 473)	Tag for data object containing structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_AUTHENTICATION_DATA (see page 474)	Tag for template nesting authentication data and parameters as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_BANK_IDENTIFIER_CODE (see page 474)	Tag for data object containing a universal Bank Identifier Code for use in automated processing in banking and related financial environments (see ISO 9362) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_BIOMETRIC_DATA_TEMPLATE (see page 474)	Template nesting biometric reference data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_BIOMETRIC_INFORMATION_GROUP_TEMPLATE (see page 474)	Tag for template used for nesting biometric information templates as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE (see page 474)	Tag for template nesting biometric information data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_CAPABILITIES (see page 475)	Tag for data element fixing card behaviours as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_DATA (see page 475)	Tag for template nesting data related to the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_EFFECTIVE_DATE (see page 475)	Tag for data object containing date from which the card can be used, under the responsibility of the card issuer as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_EXPIRATION_DATE (see page 475)	Tag for data object containing date after which the card expires as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_CARD_ISSUERS_DATA (see page 475)	Tag for data object containing proprietary card issuer's data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_SEQUENCE_NUMBER (see page 475)	Tag for data object containing number distinguishing between separate cards with the same primary account number as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_SERVICE_DATA (see page 476)	Tag for data object containing indication of methods available in the card for supporting services as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_TRACK_1 (see page 476)	Tag for structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 1 of the magnetic stripe, including discretionary data.
SCARD_TAG_CARD_TRACK_2 (see page 476)	Tag for structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 2 of the magnetic stripe, including discretionary data.
SCARD_TAG_CARD_TRACK_3 (see page 476)	Tag for structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 3 of the magnetic stripe, including discretionary data.
SCARD_TAG_CARDHOLDER_BIOMETRIC_DATA (see page 476)	Tag for data object containing biometric data related to the cardholder as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_CERTIFICATE (see page 477)	Tag for template nesting the cardholder public key, further information, signature of certification authority as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_CARDHOLDER_HANDWRITTEN_SIGNATURE_IMAGE (see page 477)	Tag for data object containing image of the cardholder's handwritten signature as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_IMAGE_TEMPLATE (see page 477)	Tag for data object containing cardholder related images stored within the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_NAME (see page 477)	Tag for data object containing name of the cardholder as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_NATIONALITY (see page 477)	Tag for data object containing nationality of the cardholder as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PORTRAIT_IMAGE (see page 478)	Tag for data object containing encoded image data, used for the cardholder portrait image as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PRIVATE_KEY (see page 478)	Tag for data element containing the cardholder's private key for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PRIVATE_KEY_TEMPLATE (see page 478)	Tag for template nesting private key related data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PUBLIC_KEY (see page 478)	Tag for data element containing the cardholder's public key data elements for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PUBLIC_KEY_TEMPLATE (see page 478)	Tag for data element containing the cardholder's public key data elements for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_RELATED_DATA (see page 478)	Tag for template nesting data related to the cardholder as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_REQUIREMENTS_EXCLUDED_FEATURES (see page 479)	Tag for data element containing cardholder's requirements for excluded features e.g. cardholder is not able to use fingerprint verification as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_CARDHOLDER_REQUIREMENTS_INCLUDED_FEATURES (see page 479)	Tag for data element containing a cardholder's requirements for included features e.g. cardholder requires audio assistance from an ATM as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CERTIFICATE_CONTENT (see page 479)	Tag for data element containing the content of a certificate as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CERTIFICATE_CONTENT_TEMPLATE (see page 479)	Tag for template for nesting certificate content data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CERTIFICATE_HOLDER_AUTHORIZATION (see page 479)	Tag for data object containing a certificate holder authorization (e.g. a role identifier) may be contained in a data element or data object with tag SCARD_TAG_CERTIFICATE_HOLDER_AUTHORIZATION as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COEXISTENT_TAG_ALLOCATION_AUTHORITY (see page 480)	Tag for template used to identify a coexistent tag allocation scheme and the authority responsible for the scheme as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COMMAND_TO_PERFORM (see page 480)	Tag for data object containing command APDU as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COMPATIBLE_TAG_ALLOCATION_AUTHORITY (see page 480)	Tag for template used to identify a compatible tag allocation scheme and the authority responsible for the scheme as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COUNTRY_CODE (see page 480)	Tag for data object containing indication of a country as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COUNTRY_CODE_ALPHA2_FORMAT (see page 480)	Tag for data object containing country Code (alpha 2 format). Indication of a country (see ISO 3166-1) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_COUNTRY_CODE_ALPHA3_FORMAT (see page 481)	Tag for data object containing country Code (alpha 3 format). Indication of a country (see ISO 3166-1) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.

SCARD_TAG_COUNTRY_CODE_AND_NATIONAL_DATA (see page 481)	Tag for data object containing indication of a country followed by national data (coding and registration defined in ISO 3166-1) and optional national data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CURRENCY_CODE (see page 481)	Tag for data object containing code for the representation of currencies and funds as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CURRENCY_EXPONENT (see page 481)	Tag for data object containing number by which an amount of the currency indicated in the card shall be multiplied as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DATE_OF_BIRTH (see page 481)	Tag for data object containing date of birth of related individual as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DEPRECATED (see page 482)	Tag for data object that is deprecated in ISO/IEC 7816-6:2004.
SCARD_TAG_DIGITAL_SIGNATURE (see page 482)	Tag for data element containing a digital signature (asymmetric or symmetric algorithm) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DIGITAL_SIGNATURE_BLOCK (see page 482)	Tag for template nesting digital signature related data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DISCRETIONARY_DATA (see page 482)	Tag for data element not defined in ISO/IEC 7816-6:2004.
SCARD_TAG_DISCRETIONARY_DATA_OBJECTS (see page 482)	Tag for data object containing concatenation of data objects not defined in ISO/IEC 7816-6:2004.
SCARD_TAG_DISPLAY_CONTROL (see page 482)	Tag for template used to control data displayed at the terminal as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DISPLAY_MESSAGE (see page 483)	Tag for data element containing a message to display as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DYNAMIC_AUTHENTICATION_TEMPLATE (see page 483)	Tag for template used in the command and response data fields of the GENERAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_DYNAMIC_EXTERNAL_AUTHENTICATION (see page 483)	Tag for composite data element used for identifying the algorithm and the key to use in the EXTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DYNAMIC_INTERNAL_AUTHENTICATION (see page 483)	Tag for composite data element used for identifying the algorithm and the key to use in the INTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DYNAMIC_MUTUAL_AUTHENTICATION (see page 483)	Tag for composite data element used for identifying the algorithm and the key to use in the mutual authentication process as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_ELEMENT_LIST (see page 484)	Tag for data object containing sequence of elements and related information, without identifiers (to be used only within a wrapper) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_EXTENDED_HEADER_LIST (see page 484)	Tag for data element for indirectly referencing data elements as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_FCI_TEMPLATE (see page 484)	Tag for template for nesting file control parameters and file management data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_FCP_TEMPLATE (see page 484)	Tag for template for nesting file control parameters as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_FILE_REFERENCE (see page 484)	Tag for data object containing reference to a file e.g. a path as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_FMD_TEMPLATE (see page 484)	Tag for template for nesting file management data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_HEADER_LIST (see page 485)	Tag for data object containing concatenation of pairs of tag fields and length fields without delimitation as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_HISTORICAL_BYTES (see page 485)	Tag for data object used to indicate operating characteristics of the card as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_INITIAL_ACCESS_DATA (see page 485)	Tag for data object containing indication of a command-to-perform for retrieving the initial data string as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTEGRATED_CIRCUIT_MANUFACTURER_IDENTIFIER (see page 485)	Tag for data object containing indication of a manufacturer of integrated circuits as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTERCHANGE_CONTROL (see page 485)	Tag for data object containing indication to use in association with a country code to indicate whether international interchange is permitted on a card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTERCHANGE_PROFILE (see page 486)	Tag for data element describing capabilities available in the card to perform an interchange transaction as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTERINDUSTRY_DATA_OBJECTS (see page 486)	Tag for template nesting interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTERNATIONAL_BANK_ACCOUNT_NUMBER (see page 486)	Tag for data object containing a number used internationally to uniquely identify the account of a customer at a financial institution (see ISO 13616) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_ISSUER_IDENTIFICATION_NUMBER (see page 486)	Tag for data element for identifying the card issuer (coding and registration defined in ISO/IEC 7812-1), possibly followed by more data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_LANGUAGE_PREFERENCES (see page 486)	Tag for data object containing indication, in order of preference, of up to four languages for the cardholder (see ISO 639) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_LOGIN_DATA_PROPRIETARY (see page 486)	Tag for data object containing proprietary information intended for connecting the interface device to a remote host, a remote server or an application within these devices as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_LOGIN_TEMPLATE (see page 487)	Tag for template conveying data intended for connecting the interface device to a remote server or an application within such devices as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_MESSAGE_REFERENCE (see page 487)	Tag for data element specifying the reference of a message as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NAME (see page 487)	Tag for data object containing name of an individual (structure and coding defined in ISO/IEC 7501-1) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_70 (see page 487)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_71 (see page 487)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_72 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_74 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_75 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_76 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_77 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_OFFSET_DATA_OBJECT (see page 488)	Tag for data object for use with commands using an odd INS code as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_PIN_USAGE_POLICY (see page 489)	Tag for data object containing indication whether PIN entry is required and under what circumstances as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_PRE_ISSUING_DATA (see page 489)	Tag for data object containing proprietary pre-issuing data as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_PRIMARY_ACCOUNT_NUMBER (see page 489)	Tag for data object containing number identifying a customer account or card (structure defined in ISO/IEC 7812 and coding in ISO 8583) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_PUBLIC_KEY_OF_CERTIFICATION_AUTHORITY (see page 489)	Tag for data element containing the certification authority's public key for digital signature functionality used to verify certificates as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_QUALIFIED_NAME (see page 489)	Tag for template nesting the name of an individual and related information, e.g., sex, date of birth etc as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SECURE_MESSAGING_TEMPLATE (see page 490)	Tag for template nesting secure messaging data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SECURITY_ENVIRONMENT_TEMPLATE (see page 490)	Tag for template nesting components of a security environment as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SECURITY_SUPPORT_TEMPLATE (see page 490)	Tag for template for encapsulating counters and auxiliary data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SERVICE_CODE (see page 490)	Tag for data object containing identification of geographic / service availability (structure defined in ISO/IEC 7813 and coding in ISO 8583) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SEX (see page 490)	Tag for data object containing gender of an individual (see ISO 5218) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SPECIAL_USER_REQUIREMENTS (see page 490)	Tag for template containing at least a tag allocation authority (tag '06', '41', '42' or '4F') and a data object by which this authority indicates the user requirements, possibly related to a disability as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION (see page 491)	Tag for data element containing a digital signature value which may be used either alone or in conjunction with the tags SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA (see page 491) and SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA (see page 491) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA (see page 491)	Tag for data object containing public key certificate data element to use either alone or in conjunction with the tag SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA (see page 491), to derive a public key value as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA (see page 491)	Tag for data object containing data auxiliary to the public key certificate, tag SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA (see page 491), used to derive the notarised public key as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_STATUS_INFORMATION (see page 491)	Tag for data object containing information on card life cycle status and processing status as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_TAG_LIST (see page 492)	Tag for data object containing concatenation of tag fields without delimitation as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_TIMER (see page 492)	Tag for data element specifying the maximal time, in tenths of a second, for performing or executing a process as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_TRANSACTION_COUNTER (see page 492)	Tag for data object containing counter incremented under the control of the application in the card after each transaction as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_TRANSACTION_DATE (see page 492)	Tag for data object containing used to recognise the date and time of the last transaction. Length is 4 for YDDD and 10 for full field as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_UNIFORM_RESOURCE_LOCATOR (see page 492)	Tag for data object containing uniform resource locator (URL, as defined in RFC 1738 and 2396) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_WRAPPER (see page 493)	Tag for template for indirect referencing and retrieval of data elements as specified in ISO/IEC 7816-6:2004.

6.1.7.4.1 Macros

6.1.7.4.1.1 SCARD_TAG_ACCOUNT_TYPE Macro

Tag for data object that indicates the type of account selected by the cardholder for the transaction (see EMV for coding) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.

C++

```
#define SCARD_TAG_ACCOUNT_TYPE 0x5F57
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.2 SCARD_TAG_ADDRESS Macro

Tag for data object containing address of an individual as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_ADDRESS 0x5F42
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.3 SCARD_TAG_ANSWER_TO_RESET Macro

Tag for data object that indicates operating characteristics of the card as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_ANSWER_TO_RESET 0x5F51
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.4 SCARD_TAG_APPLICATION_EFFECTIVE_DATE Macro

Tag for data object containing date from which the application can be used, under the responsibility of the application provider as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_EFFECTIVE_DATE 0x5F25
```


Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.5 SCARD_TAG_APPLICATION_EXPIRATION_DATE Macro

Tag for data object containing date after which an application expires, under the responsibility of the application provider as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_EXPIRATION_DATE 0x5F24
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.6 SCARD_TAG_APPLICATION_FAMILY_IDENTIFIER Macro

Tag for data object containing designation of several application areas to enable global interoperability (see ISO/IEC 14443) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.

C++

```
#define SCARD_TAG_APPLICATION_FAMILY_IDENTIFIER 0x49
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.7 SCARD_TAG_APPLICATION_IDENTIFIER Macro

Tag for data object containing data element identifying an application in the card as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_IDENTIFIER 0x4F
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.8 SCARD_TAG_APPLICATION_IMAGE Macro

Tag for data object containing icon or logo associated with an application as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_IMAGE 0x5F44
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.9 SCARD_TAG_APPLICATION_IMAGE_TEMPLATE Macro

Tag of template nesting at least an application image as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_IMAGE_TEMPLATE 0x6D
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.10 SCARD_TAG_APPLICATION_LABEL Macro

Tag for data element for use at the man-machine interface as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_LABEL 0x50
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.11 SCARD_TAG_APPLICATION_RELATED_DATA Macro

Tag for template nesting parameters of an application as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_RELATED_DATA 0x6E
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.12 SCARD_TAG_APPLICATION_TEMPLATE Macro

Tag for template identifying an application in the card as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_TEMPLATE 0x61
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.13 SCARD_TAG_APPLICATION_TRACK_1 Macro

Tag for data object containing structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_TRACK_1 0x56
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.14 SCARD_TAG_APPLICATION_TRACK_2 Macro

Tag for data object containing structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_TRACK_2 0x57
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.15 SCARD_TAG_APPLICATION_TRACK_3 Macro

Tag for data object containing structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_APPLICATION_TRACK_3 0x58
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.16 SCARD_TAG_AUTHENTICATION_DATA Macro

Tag for template nesting authentication data and parameters as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_AUTHENTICATION_DATA 0x67
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.17 SCARD_TAG_BANK_IDENTIFIER_CODE Macro

Tag for data object containing a universal Bank Identifier Code for use in automated processing in banking and related financial environments (see ISO 9362) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.

C++

```
#define SCARD_TAG_BANK_IDENTIFIER_CODE 0x5F54
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.18 SCARD_TAG_BIOMETRIC_DATA_TEMPLATE Macro

Template nesting biometric reference data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_BIOMETRIC_DATA_TEMPLATE 0x7F2E
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.19 SCARD_TAG_BIOMETRIC_INFORMATION_GROUP_TEMPLATE Macro

Tag for template used for nesting biometric information templates as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_BIOMETRIC_INFORMATION_GROUP_TEMPLATE 0x7F61
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.20 SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE Macro

Tag for template nesting biometric information data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE 0x7F60
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.21 SCARD_TAG_CARD_CAPABILITIES Macro

Tag for data element fixing card behaviours as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARD_CAPABILITIES 0x47
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.22 SCARD_TAG_CARD_DATA Macro

Tag for template nesting data related to the card as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARD_DATA 0x66
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.23 SCARD_TAG_CARD_EFFECTIVE_DATE Macro

Tag for data object containing date from which the card can be used, under the responsibility of the card issuer as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARD_EFFECTIVE_DATE 0x5F26
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.24 SCARD_TAG_CARD_EXPIRATION_DATE Macro

Tag for data object containing date after which the card expires as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARD_EXPIRATION_DATE 0x59
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.25 SCARD_TAG_CARD_ISSUERS_DATA Macro

Tag for data object containing proprietary card issuer's data as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARD_ISSUERS_DATA 0x45
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.26 SCARD_TAG_CARD_SEQUENCE_NUMBER Macro

Tag for data object containing number distinguishing between separate cards with the same primary account number as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARD_SEQUENCE_NUMBER 0x5F34
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.27 SCARD_TAG_CARD_SERVICE_DATA Macro

Tag for data object containing indication of methods available in the card for supporting services as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARD_SERVICE_DATA 0x43
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.28 SCARD_TAG_CARD_TRACK_1 Macro

Tag for structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 1 of the magnetic stripe, including discretionary data.

C++

```
#define SCARD_TAG_CARD_TRACK_1 0x5F21
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.29 SCARD_TAG_CARD_TRACK_2 Macro

Tag for structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 2 of the magnetic stripe, including discretionary data.

C++

```
#define SCARD_TAG_CARD_TRACK_2 0x5F22
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.30 SCARD_TAG_CARD_TRACK_3 Macro

Tag for structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 3 of the magnetic stripe, including discretionary data.

C++

```
#define SCARD_TAG_CARD_TRACK_3 0x5F23
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.31 SCARD_TAG_CARDHOLDER_BIOMETRIC_DATA Macro

Tag for data object containing biometric data related to the cardholder as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_BIOMETRIC_DATA 0x5F2E
```

ModuleNSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.32 SCARD_TAG_CARDHOLDER_CERTIFICATE Macro

Tag for template nesting the cardholder public key, further information, signature of certification authority as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_CERTIFICATE 0x7F21
```

ModuleNSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.33 SCARD_TAG_CARDHOLDER_HANDWRITTEN_SIGNATURE_IMAGE Macro

Tag for data object containing image of the cardholder's handwritten signature as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_HANDWRITTEN_SIGNATURE_IMAGE 0x5F43
```

ModuleNSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.34 SCARD_TAG_CARDHOLDER_IMAGE_TEMPLATE Macro

Tag for data object containing cardholder related images stored within the card as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_IMAGE_TEMPLATE 0x6C
```

ModuleNSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.35 SCARD_TAG_CARDHOLDER_NAME Macro

Tag for data object containing name of the cardholder as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_NAME 0x5F20
```

ModuleNSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.36 SCARD_TAG_CARDHOLDER_NATIONALITY Macro

Tag for data object containing nationality of the cardholder as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_NATIONALITY 0x5F2C
```

ModuleNSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.37 SCARD_TAG_CARDHOLDER_PORTRAIT_IMAGE Macro

Tag for data object containing encoded image data, used for the cardholder portrait image as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_PORTRAIT_IMAGE 0x5F40
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.38 SCARD_TAG_CARDHOLDER_PRIVATE_KEY Macro

Tag for data element containing the cardholder's private key for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_PRIVATE_KEY 0x5F48
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.39 SCARD_TAG_CARDHOLDER_PRIVATE_KEY_TEMPLATE Macro

Tag for template nesting private key related data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_PRIVATE_KEY_TEMPLATE 0x7F48
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.40 SCARD_TAG_CARDHOLDER_PUBLIC_KEY Macro

Tag for data element containing the cardholder's public key data elements for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_PUBLIC_KEY 0x5F49
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.41 SCARD_TAG_CARDHOLDER_PUBLIC_KEY_TEMPLATE Macro

Tag for data element containing the cardholder's public key data elements for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_PUBLIC_KEY_TEMPLATE 0x7F49
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.42 SCARD_TAG_CARDHOLDER_RELATED_DATA Macro

Tag for template nesting data related to the cardholder as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_RELATED_DATA 0x65
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.43 SCARD_TAG_CARDHOLDER_REQUIREMENTS_EXCLUDED_FEATURES

Macro

Tag for data element containing cardholder's requirements for excluded features e.g. cardholder is not able to use fingerprint verification as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_REQUIREMENTS_EXCLUDED_FEATURES 0x7F23
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.44 SCARD_TAG_CARDHOLDER_REQUIREMENTS_INCLUDED_FEATURES

Macro

Tag for data element containing a cardholder's requirements for included features e.g. cardholder requires audio assistance from an ATM as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CARDHOLDER_REQUIREMENTS_INCLUDED_FEATURES 0x7F22
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.45 SCARD_TAG_CERTIFICATE_CONTENT Macro

Tag for data element containing the content of a certificate as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CERTIFICATE_CONTENT 0x5F4E
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.46 SCARD_TAG_CERTIFICATE_CONTENT_TEMPLATE Macro

Tag for template for nesting certificate content data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CERTIFICATE_CONTENT_TEMPLATE 0x7F4E
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.47 SCARD_TAG_CERTIFICATE HOLDER_AUTHORIZATION Macro

Tag for data object containing a certificate holder authorization (e.g. a role identifier) may be contained in a data element or data object with tag SCARD_TAG_CERTIFICATE HOLDER_AUTHORIZATION as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CERTIFICATE HOLDER_AUTHORIZATION 0x5F4C
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.48 SCARD_TAG_COEXISTENT_TAG_ALLOCATION_AUTHORITY Macro

Tag for template used to identify a coexistent tag allocation scheme and the authority responsible for the scheme as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_COEXISTENT_TAG_ALLOCATION_AUTHORITY 0x79
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.49 SCARD_TAG_COMMAND_TO_PERFORM Macro

Tag for data object containing command APDU as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_COMMAND_TO_PERFORM 0x52
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.50 SCARD_TAG_COMPATIBLE_TAG_ALLOCATION_AUTHORITY Macro

Tag for template used to identify a compatible tag allocation scheme and the authority responsible for the scheme as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_COMPATIBLE_TAG_ALLOCATION_AUTHORITY 0x78
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.51 SCARD_TAG_COUNTRY_CODE Macro

Tag for data object containing indication of a country as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_COUNTRY_CODE 0x5F28
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.52 SCARD_TAG_COUNTRY_CODE_ALPHA2_FORMAT Macro

Tag for data object containing country Code (alpha 2 format). Indication of a country (see ISO 3166-1) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.

C++

```
#define SCARD_TAG_COUNTRY_CODE_ALPHA2_FORMAT 0x5F55
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.53 SCARD_TAG_COUNTRY_CODE_ALPHA3_FORMAT Macro

Tag for data object containing country Code (alpha 3 format). Indication of a country (see ISO 3166-1) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.

C++

```
#define SCARD_TAG_COUNTRY_CODE_ALPHA3_FORMAT 0x5F56
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.54 SCARD_TAG_COUNTRY_CODE_AND_NATIONAL_DATA Macro

Tag for data object containing indication of a country followed by national data (coding and registration defined in ISO 3166-1) and optional national data as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_COUNTRY_CODE_AND_NATIONAL_DATA 0x41
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.55 SCARD_TAG_CURRENCY_CODE Macro

Tag for data object containing code for the representation of currencies and funds as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CURRENCY_CODE 0x5F2A
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.56 SCARD_TAG_CURRENCY_EXPONENT Macro

Tag for data object containing number by which an amount of the currency indicated in the card shall be multiplied as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_CURRENCY_EXPONENT 0x5F36
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.57 SCARD_TAG_DATE_OF_BIRTH Macro

Tag for data object containing date of birth of related individual as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DATE_OF_BIRTH 0x5F2B
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.58 SCARD_TAG_DEPRECATED Macro

Tag for data object that is deprecated in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DEPRECATED 0x5F4B
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.59 SCARD_TAG_DIGITAL_SIGNATURE Macro

Tag for data element containing a digital signature (asymmetric or symmetric algorithm) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DIGITAL_SIGNATURE 0x5F3D
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.60 SCARD_TAG_DIGITAL_SIGNATURE_BLOCK Macro

Tag for template nesting digital signature related data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DIGITAL_SIGNATURE_BLOCK 0x7F3D
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.61 SCARD_TAG_DISCRETIONARY_DATA Macro

Tag for data element not defined in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DISCRETIONARY_DATA 0x53
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.62 SCARD_TAG_DISCRETIONARY_DATA_OBJECTS Macro

Tag for data object containing concatenation of data objects not defined in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DISCRETIONARY_DATA_OBJECTS 0x73
```

Module

NSmartCardsDataElements Module ([↗](#) see page 459)

6.1.7.4.1.63 SCARD_TAG_DISPLAY_CONTROL Macro

Tag for template used to control data displayed at the terminal as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DISPLAY_CONTROL 0x7F20
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.64 SCARD_TAG_DISPLAY_MESSAGE Macro

Tag for data element containing a message to display as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DISPLAY_MESSAGE 0x5F45
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.65 SCARD_TAG_DYNAMIC_AUTHENTICATION_TEMPLATE Macro

Tag for template used in the command and response data fields of the GENERAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DYNAMIC_AUTHENTICATION_TEMPLATE 0x7C
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.66 SCARD_TAG_DYNAMIC_EXTERNAL_AUTHENTICATION Macro

Tag for composite data element used for identifying the algorithm and the key to use in the EXTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DYNAMIC_EXTERNAL_AUTHENTICATION 0x5F3B
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.67 SCARD_TAG_DYNAMIC_INTERNAL_AUTHENTICATION Macro

Tag for composite data element used for identifying the algorithm and the key to use in the INTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DYNAMIC_INTERNAL_AUTHENTICATION 0x5F3A
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.68 SCARD_TAG_DYNAMIC_MUTUAL_AUTHENTICATION Macro

Tag for composite data element used for identifying the algorithm and the key to use in the mutual authentication process as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_DYNAMIC_MUTUAL_AUTHENTICATION 0x5F3C
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.69 SCARD_TAG_ELEMENT_LIST Macro

Tag for data object containing sequence of elements and related information, without identifiers (to be used only within a wrapper) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_ELEMENT_LIST 0x5F41
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.70 SCARD_TAG_EXTENDED_HEADER_LIST Macro

Tag for data element for indirectly referencing data elements as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_EXTENDED_HEADER_LIST 0x4D
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.71 SCARD_TAG_FCI_TEMPLATE Macro

Tag for template for nesting file control parameters and file management data as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_FCI_TEMPLATE 0x6F
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.72 SCARD_TAG_FCP_TEMPLATE Macro

Tag for template for nesting file control parameters as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_FCP_TEMPLATE 0x62
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.73 SCARD_TAG_FILE_REFERENCE Macro

Tag for data object containing reference to a file e.g. a path as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_FILE_REFERENCE 0x51
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.74 SCARD_TAG_FMD_TEMPLATE Macro

Tag for template for nesting file management data as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_FMD_TEMPLATE 0x64
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.75 SCARD_TAG_HEADER_LIST Macro

Tag for data object containing concatenation of pairs of tag fields and length fields without delimitation as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_HEADER_LIST 0x5D
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.76 SCARD_TAG_HISTORICAL_BYTES Macro

Tag for data object used to indicate operating characteristics of the card as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_HISTORICAL_BYTES 0x5F52
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.77 SCARD_TAG_INITIAL_ACCESS_DATA Macro

Tag for data object containing indication of a command-to-perform for retrieving the initial data string as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_INITIAL_ACCESS_DATA 0x44
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.78 SCARD_TAG_INTEGRATED_CIRCUIT_MANUFACTURER_IDENTIFIER Macro

Tag for data object containing indication of a manufacturer of integrated circuits as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_INTEGRATED_CIRCUIT_MANUFACTURER_IDENTIFIER 0x5F4D
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.79 SCARD_TAG_INTERCHANGE_CONTROL Macro

Tag for data object containing indication to use in association with a country code to indicate whether international interchange is permitted on a card as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_INTERCHANGE_CONTROL 0x5F27
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.80 SCARD_TAG_INTERCHANGE_PROFILE Macro

Tag for data element describing capabilities available in the card to perform an interchange transaction as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_INTERCHANGE_PROFILE 0x5F29
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.81 SCARD_TAG_INTERINDUSTRY_DATA_OBJECTS Macro

Tag for template nesting interindustry data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_INTERINDUSTRY_DATA_OBJECTS 0x7E
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.82 SCARD_TAG_INTERNATIONAL_BANK_ACCOUNT_NUMBER Macro

Tag for data object containing a number used internationally to uniquely identify the account of a customer at a financial institution (see ISO 13616) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.

C++

```
#define SCARD_TAG_INTERNATIONAL_BANK_ACCOUNT_NUMBER 0x5F53
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.83 SCARD_TAG_ISSUER_IDENTIFICATION_NUMBER Macro

Tag for data element for identifying the card issuer (coding and registration defined in ISO/IEC 7812-1), possibly followed by more data as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_ISSUER_IDENTIFICATION_NUMBER 0x42
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.84 SCARD_TAG_LANGUAGE_PREFERENCES Macro

Tag for data object containing indication, in order of preference, of up to four languages for the cardholder (see ISO 639) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_LANGUAGE_PREFERENCES 0x5F2D
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.85 SCARD_TAG_LOGIN_DATA_PROPRIETARY Macro

Tag for data object containing proprietary information intended for connecting the interface device to a remote host, a remote

server or an application within these devices as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_LOGIN_DATA_PROPRIETARY 0x5E
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.86 SCARD_TAG_LOGIN_TEMPLATE Macro

Tag for template conveying data intended for connecting the interface device to a remote server or an application within such devices as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_LOGIN_TEMPLATE 0x6A
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.87 SCARD_TAG_MESSAGE_REFERENCE Macro

Tag for data element specifying the reference of a message as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_MESSAGE_REFERENCE 0x5F47
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.88 SCARD_TAG_NAME Macro

Tag for data object containing name of an individual (structure and coding defined in ISO/IEC 7501-1) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_NAME 0x5B
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.89 SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_70 Macro

Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_70 0x70
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.1.90 SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_71 Macro

Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_71 0x71
```


Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.91 SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_72 Macro

Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_72 0x72
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.92 SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_74 Macro

Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_74 0x74
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.93 SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_75 Macro

Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_75 0x75
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.94 SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_76 Macro

Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_76 0x76
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.95 SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_77 Macro

Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_77 0x77
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.96 SCARD_TAG_OFFSET_DATA_OBJECT Macro

Tag for data object for use with commands using an odd INS code as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_OFFSET_DATA_OBJECT 0x54
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.97 SCARD_TAG_PIN_USAGE_POLICY Macro

Tag for data object containing indication whether PIN entry is required and under what circumstances as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_PIN_USAGE_POLICY 0x5F2F
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.98 SCARD_TAG_PRE_ISSUING_DATA Macro

Tag for data object containing proprietary pre-issuing data as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_PRE_ISSUING_DATA 0x46
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.99 SCARD_TAG_PRIMARY_ACCOUNT_NUMBER Macro

Tag for data object containing number identifying a customer account or card (structure defined in ISO/IEC 7812 and coding in ISO 8583) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_PRIMARY_ACCOUNT_NUMBER 0x5A
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.100 SCARD_TAG_PUBLIC_KEY_OF_CERTIFICATION_AUTHORITY Macro

Tag for data element containing the certification authority's public key for digital signature functionality used to verify certificates as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_PUBLIC_KEY_OF_CERTIFICATION_AUTHORITY 0x5F4A
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.101 SCARD_TAG_QUALIFIED_NAME Macro

Tag for template nesting the name of an individual and related information, e.g., sex, date of birth etc as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_QUALIFIED_NAME 0x6B
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.102 SCARD_TAG_SECURE_MESSAGING_TEMPLATE Macro

Tag for template nesting secure messaging data objects as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_SECURE_MESSAGING_TEMPLATE 0x7D
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.103 SCARD_TAG_SECURITY_ENVIRONMENT_TEMPLATE Macro

Tag for template nesting components of a security environment as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_SECURITY_ENVIRONMENT_TEMPLATE 0x7B
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.104 SCARD_TAG_SECURITY_SUPPORT_TEMPLATE Macro

Tag for template for encapsulating counters and auxiliary data as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_SECURITY_SUPPORT_TEMPLATE 0x7A
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.105 SCARD_TAG_SERVICE_CODE Macro

Tag for data object containing identification of geographic / service availability (structure defined in ISO/IEC 7813 and coding in ISO 8583) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_SERVICE_CODE 0x5F30
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.106 SCARD_TAG_SEX Macro

Tag for data object containing gender of an individual (see ISO 5218) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_SEX 0x5F35
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.107 SCARD_TAG_SPECIAL_USER_REQUIREMENTS Macro

Tag for template containing at least a tag allocation authority (tag '06', '41', '42' or '4F') and a data object by which this

authority indicates the user requirements, possibly related to a disability as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_SPECIAL_USER_REQUIREMENTS 0x68
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.108 SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION Macro

Tag for data element containing a digital signature value which may be used either alone or in conjunction with the tags SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA (see page 491) and SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA (see page 491) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION 0x5F37
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.109

SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA Macro

Tag for data object containing public key certificate data element to use either alone or in conjunction with the tag SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA (see page 491), to derive a public key value as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA 0x5F38
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.110

SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA Macro

Tag for data object containing data auxiliary to the public key certificate, tag SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA (see page 491), used to derive the notarised public key as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA 0x5F39
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.111 SCARD_TAG_STATUS_INFORMATION Macro

Tag for data object containing information on card life cycle status and processing status as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_STATUS_INFORMATION 0x48
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.112 SCARD_TAG_TAG_LIST Macro

Tag for data object containing concatenation of tag fields without delimitation as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_TAG_LIST 0x5C
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.113 SCARD_TAG_TIMER Macro

Tag for data element specifying the maximal time, in tenths of a second, for performing or executing a process as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_TIMER 0x5F46
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.114 SCARD_TAG_TRANSACTION_COUNTER Macro

Tag for data object containing counter incremented under the control of the application in the card after each transaction as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_TRANSACTION_COUNTER 0x5F32
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.115 SCARD_TAG_TRANSACTION_DATE Macro

Tag for data object containing used to recognise the date and time of the last transaction. Length is 4 for YDDD and 10 for full field as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_TRANSACTION_DATE 0x5F33
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.116 SCARD_TAG_UNIFORM_RESOURCE_LOCATOR Macro

Tag for data object containing uniform resource locator (URL, as defined in RFC 1738 and 2396) as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_UNIFORM_RESOURCE_LOCATOR 0x5F50
```

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.4.1.117 SCARD_TAG_WRAPPER Macro

Tag for template for indirect referencing and retrieval of data elements as specified in ISO/IEC 7816-6:2004.

C++

```
#define SCARD_TAG_WRAPPER 0x63
```

Module

NSmartCardsDataElements Module ([see page 459](#))

6.1.7.4.2 Files

6.1.7.4.2.1 NSmartCardsDataElements.h

Specifies tags of ISO/IEC 7816-6:2004 interindustry data elements used for interchange.

Macros

Name	Description
SCARD_TAG_ACCOUNT_TYPE (see page 471)	Tag for data object that indicates the type of account selected by the cardholder for the transaction (see EMV for coding) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_ADDRESS (see page 471)	Tag for data object containing address of an individual as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_ANSWER_TO_RESET (see page 471)	Tag for data object that indicates operating characteristics of the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_EFFECTIVE_DATE (see page 471)	Tag for data object containing date from which the application can be used, under the responsibility of the application provider as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_EXPIRATION_DATE (see page 472)	Tag for data object containing date after which an application expires, under the responsibility of the application provider as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_FAMILY_IDENTIFIER (see page 472)	Tag for data object containing designation of several application areas to enable global interoperability (see ISO/IEC 14443) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_APPLICATION_IDENTIFIER (see page 472)	Tag for data object containing data element identifying an application in the card as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_APPLICATION_IMAGE (see page 472)	Tag for data object containing icon or logo associated with an application as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_IMAGE_TEMPLATE (see page 472)	Tag of template nesting at least an application image as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_LABEL (see page 473)	Tag for data element for use at the man-machine interface as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_RELATED_DATA (see page 473)	Tag for template nesting parameters of an application as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_TEMPLATE (see page 473)	Tag for template identifying an application in the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_TRACK_1 (see page 473)	Tag for data object containing structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_TRACK_2 (see page 473)	Tag for data object containing structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_APPLICATION_TRACK_3 (see page 473)	Tag for data object containing structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_AUTHENTICATION_DATA (see page 474)	Tag for template nesting authentication data and parameters as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_BANK_IDENTIFIER_CODE (see page 474)	Tag for data object containing a universal Bank Identifier Code for use in automated processing in banking and related financial environments (see ISO 9362) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.

SCARD_TAG_BIOMETRIC_DATA_TEMPLATE (see page 474)	Template nesting biometric reference data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_BIOMETRIC_INFORMATION_GROUP_TEMPLATE (see page 474)	Tag for template used for nesting biometric information templates as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE (see page 474)	Tag for template nesting biometric information data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_CAPABILITIES (see page 475)	Tag for data element fixing card behaviours as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_DATA (see page 475)	Tag for template nesting data related to the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_EFFECTIVE_DATE (see page 475)	Tag for data object containing date from which the card can be used, under the responsibility of the card issuer as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_EXPIRATION_DATE (see page 475)	Tag for data object containing date after which the card expires as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_ISSUERS_DATA (see page 475)	Tag for data object containing proprietary card issuer's data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_SEQUENCE_NUMBER (see page 475)	Tag for data object containing number distinguishing between separate cards with the same primary account number as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_SERVICE_DATA (see page 476)	Tag for data object containing indication of methods available in the card for supporting services as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARD_TRACK_1 (see page 476)	Tag for structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 1 of the magnetic stripe, including discretionary data.

SCARD_TAG_CARD_TRACK_2 (see page 476)	Tag for structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 2 of the magnetic stripe, including discretionary data.
SCARD_TAG_CARD_TRACK_3 (see page 476)	Tag for structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein as specified in ISO/IEC 7816-6:2004. The data content is the same as track 3 of the magnetic stripe, including discretionary data.
SCARD_TAG_CARDHOLDER_BIOMETRIC_DATA (see page 476)	Tag for data object containing biometric data related to the cardholder as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_CERTIFICATE (see page 477)	Tag for template nesting the cardholder public key, further information, signature of certification authority as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_HANDWRITTEN_SIGNATURE_IMAGE (see page 477)	Tag for data object containing image of the cardholder's handwritten signature as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_IMAGE_TEMPLATE (see page 477)	Tag for data object containing cardholder related images stored within the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_NAME (see page 477)	Tag for data object containing name of the cardholder as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_NATIONALITY (see page 477)	Tag for data object containing nationality of the cardholder as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PORTRAIT_IMAGE (see page 478)	Tag for data object containing encoded image data, used for the cardholder portrait image as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_CARDHOLDER_PRIVATE_KEY (see page 478)	Tag for data element containing the cardholder's private key for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PRIVATE_KEY_TEMPLATE (see page 478)	Tag for template nesting private key related data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PUBLIC_KEY (see page 478)	Tag for data element containing the cardholder's public key data elements for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_PUBLIC_KEY_TEMPLATE (see page 478)	Tag for data element containing the cardholder's public key data elements for digital signature functionality using asymmetric mechanisms as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_RELATED_DATA (see page 478)	Tag for template nesting data related to the cardholder as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_REQUIREMENTS_EXCLUDED_FEATURES (see page 479)	Tag for data element containing cardholder's requirements for excluded features e.g. cardholder is not able to use fingerprint verification as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CARDHOLDER_REQUIREMENTS_INCLUDED_FEATURES (see page 479)	Tag for data element containing a cardholder's requirements for included features e.g. cardholder requires audio assistance from an ATM as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CERTIFICATE_CONTENT (see page 479)	Tag for data element containing the content of a certificate as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CERTIFICATE_CONTENT_TEMPLATE (see page 479)	Tag for template for nesting certificate content data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CERTIFICATE_HOLDER_AUTHORIZATION (see page 479)	Tag for data object containing a certificate holder authorization (e.g. a role identifier) may be contained in a data element or data object with tag SCARD_TAG_CERTIFICATE_HOLDER_AUTHORIZATION as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_COEXISTENT_TAG_ALLOCATION_AUTHORITY (see page 480)	Tag for template used to identify a coexistent tag allocation scheme and the authority responsible for the scheme as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COMMAND_TO_PERFORM (see page 480)	Tag for data object containing command APDU as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COMPATIBLE_TAG_ALLOCATION_AUTHORITY (see page 480)	Tag for template used to identify a compatible tag allocation scheme and the authority responsible for the scheme as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COUNTRY_CODE (see page 480)	Tag for data object containing indication of a country as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_COUNTRY_CODE_ALPHA2_FORMAT (see page 480)	Tag for data object containing country Code (alpha 2 format). Indication of a country (see ISO 3166-1) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_COUNTRY_CODE_ALPHA3_FORMAT (see page 481)	Tag for data object containing country Code (alpha 3 format). Indication of a country (see ISO 3166-1) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_COUNTRY_CODE_AND_NATIONAL_DATA (see page 481)	Tag for data object containing indication of a country followed by national data (coding and registration defined in ISO 3166-1) and optional national data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CURRENCY_CODE (see page 481)	Tag for data object containing code for the representation of currencies and funds as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_CURRENCY_EXPONENT (see page 481)	Tag for data object containing number by which an amount of the currency indicated in the card shall be multiplied as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DATE_OF_BIRTH (see page 481)	Tag for data object containing date of birth of related individual as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DEPRECATED (see page 482)	Tag for data object that is deprecated in ISO/IEC 7816-6:2004.

SCARD_TAG_DIGITAL_SIGNATURE (see page 482)	Tag for data element containing a digital signature (asymmetric or symmetric algorithm) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DIGITAL_SIGNATURE_BLOCK (see page 482)	Tag for template nesting digital signature related data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DISCRETIONARY_DATA (see page 482)	Tag for data element not defined in ISO/IEC 7816-6:2004.
SCARD_TAG_DISCRETIONARY_DATA_OBJECTS (see page 482)	Tag for data object containing concatenation of data objects not defined in ISO/IEC 7816-6:2004.
SCARD_TAG_DISPLAY_CONTROL (see page 482)	Tag for template used to control data displayed at the terminal as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DISPLAY_MESSAGE (see page 483)	Tag for data element containing a message to display as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DYNAMIC_AUTHENTICATION_TEMPLATE (see page 483)	Tag for template used in the command and response data fields of the GENERAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DYNAMIC_EXTERNAL_AUTHENTICATION (see page 483)	Tag for composite data element used for identifying the algorithm and the key to use in the EXTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DYNAMIC_INTERNAL_AUTHENTICATION (see page 483)	Tag for composite data element used for identifying the algorithm and the key to use in the INTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_DYNAMIC_MUTUAL_AUTHENTICATION (see page 483)	Tag for composite data element used for identifying the algorithm and the key to use in the mutual authentication process as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_ELEMENT_LIST (see page 484)	Tag for data object containing sequence of elements and related information, without identifiers (to be used only within a wrapper) as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_EXTENDED_HEADER_LIST (see page 484)	Tag for data element for indirectly referencing data elements as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_FCI_TEMPLATE (see page 484)	Tag for template for nesting file control parameters and file management data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_FCP_TEMPLATE (see page 484)	Tag for template for nesting file control parameters as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_FILE_REFERENCE (see page 484)	Tag for data object containing reference to a file e.g. a path as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_FMD_TEMPLATE (see page 484)	Tag for template for nesting file management data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_HEADER_LIST (see page 485)	Tag for data object containing concatenation of pairs of tag fields and length fields without delimitation as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_HISTORICAL_BYTES (see page 485)	Tag for data object used to indicate operating characteristics of the card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INITIAL_ACCESS_DATA (see page 485)	Tag for data object containing indication of a command-to-perform for retrieving the initial data string as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTEGRATED_CIRCUIT_MANUFACTURER_IDENTIFIER (see page 485)	Tag for data object containing indication of a manufacturer of integrated circuits as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTERCHANGE_CONTROL (see page 485)	Tag for data object containing indication to use in association with a country code to indicate whether international interchange is permitted on a card as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTERCHANGE_PROFILE (see page 486)	Tag for data element describing capabilities available in the card to perform an interchange transaction as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_INTERINDUSTRY_DATA_OBJECTS (see page 486)	Tag for template nesting interindustry data objects as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_INTERNATIONAL_BANK_ACCOUNT_NUMBER (see page 486)	Tag for data object containing a number used internationally to uniquely identify the account of a customer at a financial institution (see ISO 13616) as specified in ISO/IEC 7816-6:2004/Cor 1:2006.
SCARD_TAG_ISSUER_IDENTIFICATION_NUMBER (see page 486)	Tag for data element for identifying the card issuer (coding and registration defined in ISO/IEC 7812-1), possibly followed by more data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_LANGUAGE_PREFERENCES (see page 486)	Tag for data object containing indication, in order of preference, of up to four languages for the cardholder (see ISO 639) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_LOGIN_DATA_PROPRIETARY (see page 486)	Tag for data object containing proprietary information intended for connecting the interface device to a remote host, a remote server or an application within these devices as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_LOGIN_TEMPLATE (see page 487)	Tag for template conveying data intended for connecting the interface device to a remote server or an application within such devices as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_MESSAGE_REFERENCE (see page 487)	Tag for data element specifying the reference of a message as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NAME (see page 487)	Tag for data object containing name of an individual (structure and coding defined in ISO/IEC 7501-1) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_70 (see page 487)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_71 (see page 487)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_72 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_74 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_75 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_76 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_77 (see page 488)	Tag for template nesting non-interindustry data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_OFFSET_DATA_OBJECT (see page 488)	Tag for data object for use with commands using an odd INS code as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_PIN_USAGE_POLICY (see page 489)	Tag for data object containing indication whether PIN entry is required and under what circumstances as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_PRE_ISSUING_DATA (see page 489)	Tag for data object containing proprietary pre-issuing data as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_PRIMARY_ACCOUNT_NUMBER (see page 489)	Tag for data object containing number identifying a customer account or card (structure defined in ISO/IEC 7812 and coding in ISO 8583) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_PUBLIC_KEY_OF_CERTIFICATION_AUTHORITY (see page 489)	Tag for data element containing the certification authority's public key for digital signature functionality used to verify certificates as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_QUALIFIED_NAME (see page 489)	Tag for template nesting the name of an individual and related information, e.g., sex, date of birth etc as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SECURE_MESSAGING_TEMPLATE (see page 490)	Tag for template nesting secure messaging data objects as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SECURITY_ENVIRONMENT_TEMPLATE (see page 490)	Tag for template nesting components of a security environment as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SECURITY_SUPPORT_TEMPLATE (see page 490)	Tag for template for encapsulating counters and auxiliary data as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_SERVICE_CODE (see page 490)	Tag for data object containing identification of geographic / service availability (structure defined in ISO/IEC 7813 and coding in ISO 8583) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SEX (see page 490)	Tag for data object containing gender of an individual (see ISO 5218) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_SPECIAL_USER_REQUIREMENTS (see page 490)	Tag for template containing at least a tag allocation authority (tag '06', '41', '42' or '4F') and a data object by which this authority indicates the user requirements, possibly related to a disability as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION (see page 491)	Tag for data element containing a digital signature value which may be used either alone or in conjunction with the tags SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA (see page 491) and SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA (see page 491) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA (see page 491)	Tag for data object containing public key certificate data element to use either alone or in conjunction with the tag SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA (see page 491), to derive a public key value as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SECOND_ASSOCIATED_DATA (see page 491)	Tag for data object containing data auxiliary to the public key certificate, tag SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIRST_ASSOCIATED_DATA (see page 491), used to derive the notarised public key as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_STATUS_INFORMATION (see page 491)	Tag for data object containing information on card life cycle status and processing status as specified in ISO/IEC 7816-6:2004.

SCARD_TAG_TAG_LIST (see page 492)	Tag for data object containing concatenation of tag fields without delimitation as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_TIMER (see page 492)	Tag for data element specifying the maximal time, in tenths of a second, for performing or executing a process as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_TRANSACTION_COUNTER (see page 492)	Tag for data object containing counter incremented under the control of the application in the card after each transaction as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_TRANSACTION_DATE (see page 492)	Tag for data object containing used to recognise the date and time of the last transaction. Length is 4 for YDDD and 10 for full field as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_UNIFORM_RESOURCE_LOCATOR (see page 492)	Tag for data object containing uniform resource locator (URL, as defined in RFC 1738 and 2396) as specified in ISO/IEC 7816-6:2004.
SCARD_TAG_WRAPPER (see page 493)	Tag for template for indirect referencing and retrieval of data elements as specified in ISO/IEC 7816-6:2004.

Module

NSmartCardsDataElements Module (see page 459)

6.1.7.5 SimpleTlv Module

Provides functionality for creation, validation, retrieval, update and deletion of SIMPLE-TLV data objects (SimpleTlv) that are fully compliant with SIMPLE-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Files

Name	Description
SimpleTlv.h (see page 513)	Provides functionality for creation, validation, retrieval, update and deletion of SIMPLE-TLV data objects (SimpleTlv) that are fully compliant with SIMPLE-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Functions

	Name	Description
	SimpleTagsValid (see page 505)	Gets the option specifying whether SIMPLE-TLV tag field is valid, i.e. does not violate any of the encoding rules specified in ISO/IEC 7816-4:2005.
	SimpleTlvClone (see page 505)	Creates a clone of a SIMPLE-TLV data object.
	SimpleTlvCreate (see page 506)	Creates SIMPLE-TLV data object.
	SimpleTlvCreateFromMemory (see page 506)	Creates SIMPLE-TLV data object from memory buffer.

SimpleTlvGetLength (see page 507)	Gets the SIMPLE-TLV data object length field.
SimpleTlvGetSize (see page 507)	Gets the size of a SIMPLE-TLV data object.
SimpleTlvGetSizeMany (see page 508)	Gets the total size of an array of SIMPLE-TLV data objects.
SimpleTlvGetTag (see page 508)	Gets the SIMPLE-TLV data object tag field.
SimpleTlvGetUseThreeByteLengthAlways (see page 509)	Indicates whether 3 bytes size length field should always be used in the SIMPLE-TLV.
SimpleTlvGetValue (see page 509)	Gets the value field of a SIMPLE-TLV data object.
SimpleTlvGetValuePtr (see page 510)	Gets the pointer to the value field of a SIMPLE-TLV data object.
SimpleTlvLoadManyFromMemory (see page 510)	Creates an array of SIMPLE-TLV data objects from memory buffer.
SimpleTlvSaveManyToMemory (see page 511)	Writes an array of SIMPLE-TLV data objects to memory buffer.
SimpleTlvSaveToMemory (see page 511)	Writes SIMPLE-TLV data object to memory buffer.
SimpleTlvSetLength (see page 512)	Sets the length field of a SIMPLE-TLV data object and updates the size of SIMPLE-TLV respectively.
SimpleTlvSetUseThreeByteLengthAlways (see page 512)	Sets the option specifying whether 3 bytes size length field should always be used in the SIMPLE-TLV.
SimpleTlvSetValue (see page 513)	Sets the value field of a SIMPLE-TLV data object and updates the length field and the size of a SIMPLE-TLV respectively.
HSimpleTlv (see page 513)	Handle to a SIMPLE-TLV data object.

6.1.7.5.1 Functions

6.1.7.5.1.1 SimpleTagIsValid Function

Gets the option specifying whether SIMPLE-TLV tag field is valid, i.e. does not violate any of the encoding rules specified in ISO/IEC 7816-4:2005.

C++

```
NBool N_API SimpleTagIsValid(NByte value);
```

Parameters

Parameters	Description
NByte value	[in] SIMPLE-TLV tag field value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.2 SimpleTlvClone Function

Creates a clone of a SIMPLE-TLV data object.

C++

```
NResult N_API SimpleTlvClone(HSimpleTlv hSimpleTlv, HSimpleTlv * pHClonedSimpleTlv);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SimpleTlv (see page 504) object to be cloned.
HSimpleTlv * pHClonedSimpleTlv	[out] Pointer to HSimpleTlv (see page 513) that receives a handle to the clone of a SimpleTlv (see page 504) object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> or <i>pHClonedSimpleTlv</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.3 SimpleTlvCreate Function

Creates SIMPLE-TLV data object.

C++

```
NResult N_API SimpleTlvCreate(NByte tag, NBool useThreeByteLengthAlways, HSimpleTlv * pHSimpleTlv);
```

Parameters

Parameters	Description
NByte tag	[in] SIMPLE-TLV tag field value.
NBool useThreeByteLengthAlways	[in] Boolean parameter specifying whether only 3 bytes size length field should always be used in the SIMPLE-TLV.
HSimpleTlv * pHSimpleTlv	[out] Pointer to HSimpleTlv (see page 513) that receives a handle to created SIMPLE-TLV object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pHSimpleTlv</i> is NULL (see page 133).
N_E_ARGUMENT (see page 78)	<i>tag</i> is invalid.

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.4 SimpleTlvCreateFromMemory Function

Creates SIMPLE-TLV data object from memory buffer.

C++

```
NResult N_API SimpleTlvCreateFromMemory(const void * pBuffer, NSizeType bufferSize, HSimpleTlv * pHSimpleTlv);
```

Parameters

Parameters	Description
const void * pBuffer	[in] Pointer to memory buffer to read SIMPLE-TLV data from.
NSizeType bufferSize	Size of memory buffer to read SIMPLE-TLV data from.
HSimpleTlv * pHSimpleTlv	[out] Pointer to HSimpleTlv (see page 513) that receives a handle to created SIMPLE-TLV data object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	pBuffer or pHSimpleTlv is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.5 SimpleTlvGetLength Function

Gets the SIMPLE-TLV data object length field.

C++

```
NResult N_API SimpleTlvGetLength(HSimpleTlv hSimpleTlv, NSizeType * pValue);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.
NSizeType * pValue	[out] Pointer to NInt (see page 117) that receives the value of the SIMPLE-TLV length field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hSimpleTlv or pValue is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.6 SimpleTlvGetSize Function

Gets the size of a SIMPLE-TLV data object.

C++

```
NResult N_API SimpleTlvGetSize(HSimpleTlv hSimpleTlv, NSizeType * pValue);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.

NSizeType * pValue	[out] Pointer to NSizeType (see page 119) that receives the value of the SIMPLE-TLV size.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> or <i>pValue</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.7 SimpleTlvGetSizeMany Function

Gets the total size of an array of SIMPLE-TLV data objects.

C++

```
NResult N_API SimpleTlvGetSizeMany(const HSimpleTlv * arHSimpleTlvs, NInt count, NSizeType * pSize);
```

Parameters

Parameters	Description
const HSimpleTlv * arHSimpleTlvs	[in] Array of handles to SIMPLE-TLV data objects.
NInt count	[in] Number of elements in the array.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives the total size of all SIMPLE-TLV data objects from the array.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>arHSimpleTlvs</i> is NULL (see page 133) and <i>count</i> not equal zero or <i>pSize</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>count</i> is less than zero.

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.8 SimpleTlvGetTag Function

Gets the SIMPLE-TLV data object tag field.

C++

```
NResult N_API SimpleTlvGetTag(HSimpleTlv hSimpleTlv, NByte * pValue);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.
NByte * pValue	[out] Pointer to NInt (see page 117) that receives the value of the SIMPLE-TLV tag field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> or <i>pValue</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.9 SimpleTlvGetUseThreeByteLengthAlways Function

Indicates whether 3 bytes size length field should always be used in the SIMPLE-TLV.

C++

```
NResult N_API SimpleTlvGetUseThreeByteLengthAlways(HSimpleTlv hSimpleTlv, NBool * pValue);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.
NBool * pValue	[out] Pointer to NBool (see page 116) that receives parameter specifying whether only 3 bytes size length field should always be used in the SIMPLE-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> or <i>pValue</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.10 SimpleTlvGetValue Function

Gets the value field of a SIMPLE-TLV data object.

C++

```
NResult N_API SimpleTlvGetValue(HSimpleTlv hSimpleTlv, void * pValue);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.
void * pValue	[out] Pointer to void that receives the copy of the value field of a SIMPLE-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> is NULL (see page 133) or <i>pValue</i> is NULL (see page 133) and length is not equal to zero.

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.11 SimpleTlvGetValuePtr Function

Gets the pointer to the value field of a SIMPLE-TLV data object.

C++

```
NResult N_API SimpleTlvGetValuePtr(HSimpleTlv hSimpleTlv, void * * pValue);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.
void * * pValue	[out] Pointer to void * that receives the pointer to the value field of a SIMPLE-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> or <i>pValue</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.12 SimpleTlvLoadManyFromMemory Function

Creates an array of SIMPLE-TLV data objects from memory buffer.

C++

```
NResult N_API SimpleTlvLoadManyFromMemory(const void * pBuffer, NSizeType bufferSize, HSimpleTlv * * pArHSimpleTlvs, NInt * pCount);
```

Parameters

Parameters	Description
const void * pBuffer	[in] Pointer to memory buffer to read SIMPLE-TLVs data from.
NSizeType bufferSize	Size of memory buffer to read SIMPLE-TLVs data from.
HSimpleTlv * * pArHSimpleTlvs	[out] Pointer to an array of pointers to handles of created SIMPLE-TLV data objects.
NInt * pCount	[out] Pointer to NInt (see page 117) that receives the number of created SIMPLE-TLV data objects.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pBuffer</i> is NULL (see page 133) and <i>bufferSize</i> is not equal to zero or <i>pArHSimpleTlvs</i> or <i>pCount</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.13 SimpleTlvSaveManyToMemory Function

Writes an array of SIMPLE-TLV data objects to memory buffer.

C++

```
NResult N_API SimpleTlvSaveManyToMemory(const HSimpleTlv * arHSimpleTlvs, NInt count, void * pBuffer, NSizeType bufferSize, NSizeType * pSize);
```

Parameters

Parameters	Description
const HSimpleTlv * arHSimpleTlvs	[in] Pointer to an array of handles to SIMPLE-TLV data objects.
NInt count	[in] Number of elements in the array.
void * pBuffer	[out] Pointer to memory buffer to write SIMPLE-TLVs in.
NSizeType bufferSize	Size of memory buffer to write SIMPLE-TLVs in.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives total size of written SIMPLE-TLVs.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>bufferSize</i> is not big enough.
N_E_ARGUMENT_NULL (see page 78)	<i>pBuffer</i> is NULL (see page 133) and <i>bufferSize</i> is not equal zero.

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.14 SimpleTlvSaveToMemory Function

Writes SIMPLE-TLV data object to memory buffer.

C++

```
NResult N_API SimpleTlvSaveToMemory(HSimpleTlv hSimpleTlv, void * pBuffer, NSizeType bufferSize, NSizeType * pSize);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object to be written.
void * pBuffer	[out] Pointer to memory buffer to write SIMPLE-TLV in.
NSizeType bufferSize	Size of memory buffer to write SIMPLE-TLV in.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives the size of written SIMPLE-TLV data object.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>bufferSize</i> is not big enough.
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> or <i>pBuffer</i> or <i>pSize</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.15 SimpleTlvSetLength Function

Sets the length field of a SIMPLE-TLV data object and updates the size of SIMPLE-TLV respectively.

C++

```
NResult N_API SimpleTlvSetLength(HSimpleTlv hSimpleTlv, NSizeType value);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.
NSizeType value	[in] SIMPLE-TLV length field value to be set.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_OVERFLOW (see page 81)	<i>value</i> is to big.
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.16 SimpleTlvSetUseThreeByteLengthAlways Function

Sets the option specifying whether 3 bytes size length field should always be used in the SIMPLE-TLV.

C++

```
NResult N_API SimpleTlvSetUseThreeByteLengthAlways(HSimpleTlv hSimpleTlv, NBool value);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.
NBool value	[in] Boolean parameter specifying whether 3 bytes size length field should always be used in the SIMPLE-TLV.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> is NULL (see page 133).

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.17 SimpleTlvSetValue Function

Sets the value field of a SIMPLE-TLV data object and updates the length field and the size of a SIMPLE-TLV respectively.

C++

```
NResult N_API SimpleTlvSetValue(HSimpleTlv hSimpleTlv, const void * pValue, NSizeType length);
```

Parameters

Parameters	Description
HSimpleTlv hSimpleTlv	[in] Handle to a SIMPLE-TLV data object.
const void * pValue	[in] Pointer to the value to be set to the SIMPLE-TLV value field.
NSizeType length	[in] Length of the value to be set to the SIMPLE-TLV value field.

Returns

If the function succeeds, the return value is N_OK (see page 82). If the function fails, the return value is one of the following error codes:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hSimpleTlv</i> is NULL (see page 133) or <i>pValue</i> is NULL (see page 133) and length is not equal zero.

Module

SimpleTlv Module (see page 504)

6.1.7.5.1.18 HSimpleTlv

Handle to a SIMPLE-TLV data object.

Module

SimpleTlv Module (see page 504)

6.1.7.5.2 Files**6.1.7.5.2.1 SimpleTlv.h**

Provides functionality for creation, validation, retrieval, update and deletion of SIMPLE-TLV data objects (SimpleTlv (see page 504)) that are fully compliant with SIMPLE-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Functions

	Name	Description
	SimpleTagsValid (see page 505)	Gets the option specifying whether SIMPLE-TLV tag field is valid, i.e. does not violate any of the encoding rules specified in ISO/IEC 7816-4:2005.

SimpleTlvClone (see page 505)	Creates a clone of a SIMPLE-TLV data object.
SimpleTlvCreate (see page 506)	Creates SIMPLE-TLV data object.
SimpleTlvCreateFromMemory (see page 506)	Creates SIMPLE-TLV data object from memory buffer.
SimpleTlvGetLength (see page 507)	Gets the SIMPLE-TLV data object length field.
SimpleTlvGetSize (see page 507)	Gets the size of a SIMPLE-TLV data object.
SimpleTlvGetSizeMany (see page 508)	Gets the total size of an array of SIMPLE-TLV data objects.
SimpleTlvGetTag (see page 508)	Gets the SIMPLE-TLV data object tag field.
SimpleTlvGetUseThreeByteLengthAlways (see page 509)	Indicates whether 3 bytes size length field should always be used in the SIMPLE-TLV.
SimpleTlvGetValue (see page 509)	Gets the value field of a SIMPLE-TLV data object.
SimpleTlvGetValuePtr (see page 510)	Gets the pointer to the value field of a SIMPLE-TLV data object.
SimpleTlvLoadManyFromMemory (see page 510)	Creates an array of SIMPLE-TLV data objects from memory buffer.
SimpleTlvSaveManyToMemory (see page 511)	Writes an array of SIMPLE-TLV data objects to memory buffer.
SimpleTlvSaveToMemory (see page 511)	Writes SIMPLE-TLV data object to memory buffer.
SimpleTlvSetLength (see page 512)	Sets the length field of a SIMPLE-TLV data object and updates the size of SIMPLE-TLV respectively.
SimpleTlvSetUseThreeByteLengthAlways (see page 512)	Sets the option specifying whether 3 bytes size length field should always be used in the SIMPLE-TLV.
SimpleTlvSetValue (see page 513)	Sets the value field of a SIMPLE-TLV data object and updates the length field and the size of a SIMPLE-TLV respectively.

Module

SimpleTlv Module (see page 504)

6.1.8 NTemplates Library

Provides functionality for packing, unpacking and editing Neurotechnology Templates (NTemplates (see page 605)), Fingers and Palms Templates (NFTemplates), Faces Templates (NLTemplates), Iris Templates (NETemplates), Finger and Palms Records (NFRrecords), Face Records (NLRecords), Iris Records (NERRecords).

Remarks**Requirements (Windows)**

- **Import Library:** NTemplates.dll.lib
- **DLLs:**
 - NTemplates.dll
 - NCore.dll (see page 74)

Requirements (Linux)

- **Shared objects:**
 - libNTemplates.so
 - libNcore.so

Modules

Name	Description
NFRecordV1 Module (see page 515)	Provides functions for saving NFRecord to NFRecord 1.0 version.
NFRecord Module (see page 518)	Provides functionality for packing, unpacking and editing Neurotechnology Finger and Palm Records (NFRecords).
NFTemplate Module (see page 588)	Provides functionality for packing, unpacking and editing Neurotechnology Finger and Palm Templates (NFTemplates).
NTemplates Module (see page 605)	Provides functionality for retrieving information about NTemplates library.

6.1.8.1 NFRecordV1 Module

Provides functions for saving NFRecord to NFRecord 1.0 version.

Files

Name	Description
NFRecordV1.h (see page 518)	Header file for the NFRecord. Provides functions for saving NFRecord to NFRecord 1.0 version.

Functions

	Name	Description
⇒	NFRecordGetMaxSizeV1 (see page 515)	Retrieves the maximal size of a packed in version 1.0 format NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.
⇒	NFRecordGetSizeV1 (see page 516)	Calculates packed in version 1.0 format size of the NFRecord.
⇒	NFRecordSaveToMemoryV1 (see page 517)	Packs the NFRecord into the specified memory buffer in version 1.0 format.

6.1.8.1.1 Functions**6.1.8.1.1.1 NFRecordGetMaxSizeV1 Function**

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

C++

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

Parameters

Parameters	Description
NMinutiaFormat minutiaFormat	[in] The minutia format.
NInt minutiaCount	[in] The number of minutiae.
NInt coreCount	[in] The number of cores.
NInt deltaCount	[in] The number of deltas.
NInt doubleCoreCount	[in] The number of double cores.
NInt boWidth	[in] The width of blocked orientations.
NInt boHeight	[in] The height of blocked orientations.

NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives maximal size of packed NRecord.

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned:

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>minutiaFormat</i> is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>pSize</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>minutiaCount</i> is less than zero or greater than or equal to NFR_MAX_MINUTIA_COUNT (see page 579). - or - <i>coreCount</i> is less than zero or greater than or equal to NFR_MAX_CORE_COUNT (see page 581). - or - <i>deltaCount</i> is less than zero or greater than or equal to NFR_MAX_DELTA_COUNT (see page 581). - or - <i>doubleCoreCount</i> is less than zero or greater than or equal to NFR_MAX_DOUBLE_CORE_COUNT (see page 579). - or - <i>boWidth</i> or <i>boHeight</i> is less than zero or greater than or equal to NFR_MAX_BLOCKED_ORIENTS_DIMENSION.

Remarks

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

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

Module

NRecordV1 Module (see page 515)

6.1.8.1.1.2 NRecordGetSizeV1 Function

Calculates packed in version 1.0 format size of the NRecord.

C++

```
NResult N_API NRecordGetSizeV1(HNRecord hRecord, NUInt flags, NSizeType * pSize);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function. See remarks.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives size of packed NRecord.

Returns

If the function succeeds, the return value is N_OK (see page 82). Otherwise, one of the following error codes is returned:

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pSize</i> is NULL (see page 133).

Remarks

For the list of flags that are supported see `NFRecordSaveToMemoryV1` (see page 517) function.

Module

NFRecordV1 Module (see page 515)

6.1.8.1.1.3 NFRecordSaveToMemoryV1 Function

Packs the NFRecord into the specified memory buffer in version 1.0 format.

C++

```
NResult N_API NFRecordSaveToMemoryV1(HNFRecord hRecord, void * pBuffer, NSizeType bufferSize, NUInt flags, NSizeType * pSize);
```

Parameters

Parameters	Description
HNFRecord <i>hRecord</i>	[in] Handle to the NFRecord object.
void * <i>pBuffer</i>	[out] Pointer to memory buffer to store packed NFRecord. Can be NULL (see page 133).
NSizeType <i>bufferSize</i>	[in] Size of memory buffer to store packed NFRecord.
NUInt <i>flags</i>	[in] Bitwise combination of zero or more flags that controls behavior of the function. See remarks.
NSizeType * <i>pSize</i>	[out] Pointer to NSizeType (see page 119) that receives size of packed NFRecord.

Returns

If the function succeeds, the return value is `N_OK` (see page 82). Otherwise, one of the following error codes is returned:

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>buffer</i> is not NULL (see page 133) and <i>bufferSize</i> is less than size required to store packed NFRecord version 1.0
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pSize</i> is NULL (see page 133). - or - <i>buffer</i> is NULL (see page 133) and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is NULL (see page 133) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as `NFRecordGetSizeV1` (see page 516) function.

If *buffer* is not NULL (see page 133), *bufferSize* must not be less than value calculated with `NFRecordGetSizeV1` (see page 516) function.

Note that blocked orientations are not packed by default.

The following flags are supported:

- `NFR_SAVE_BLOCKED_ORIENTS` (see page 580)
- `NFR_SKIP_CURVATURES` (see page 580)
- `NFR_SKIP_GS` (see page 580)
- `NFR_SKIP_SINGULAR_POINTS` (see page 581)

Module

NFRecordV1 Module ([see page 515](#))

6.1.8.1.2 Files**6.1.8.1.2.1 NFRecordV1.h**

Header file for the NFRecord. Provides functions for saving NFRecord to NFRecord 1.0 version.

Functions

	Name	Description
	NFRecordGetMaxSizeV1 (see page 515)	Retrieves the maximal size of a packed in version 1.0 format NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.
	NFRecordGetSizeV1 (see page 516)	Calculates packed in version 1.0 format size of the NFRecord.
	NFRecordSaveToMemoryV1 (see page 517)	Packs the NFRecord into the specified memory buffer in version 1.0 format.

Module

NFRecordV1 Module ([see page 515](#))

6.1.8.2 NFRecord Module

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

Files

Name	Description
NFRecord.h (see page 583)	Header file for the NFRecord. Provides functionality for packing, unpacking and editing Neurotechnology Finger Records (NFRecords).

Functions

	Name	Description
	NFRecordAddCore (see page 523)	Adds a NFCore (see page 571) to the end of NFRecord cores.
	NFRecordGetMaxSize (see page 524)	Retrieves the maximal size of a packed NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type (for fingerprints) or palmprint values.
	NFRecordAddDelta (see page 525)	Adds a NFDelta (see page 571) to the end of NFRecord deltas.
	NFRecordAddDoubleCore (see page 526)	Adds a NFDoubleCore (see page 572) to the end of NFRecord double cores.
	NFRecordAddMinutia (see page 526)	Adds a NFMinutia (see page 573) to the end of NFRecord minutiae.
	NFRecordCheck (see page 527)	Checks if format of the packed NFRecord is correct.
	NFRecordClearCores (see page 527)	Removes all cores from the NFRecord.
	NFRecordClearDeltas (see page 527)	Removes all deltas from the NFRecord.
	NFRecordClearDoubleCores (see page 528)	Removes all double cores from the NFRecord.

◆	NFRecordClearMinutiae (see page 528)	Removes all minutiae from the NFRecord.
◆	NFRecordClone (see page 529)	Creates a copy of the NFRecord.
◆	NFRecordCreate (see page 529)	Creates an empty NFRecord.
◆	NFRecordCreateEx (see page 530)	Creates an empty NFRecord.
◆	NFRecordCreateFromMemory (see page 531)	Unpacks a NFRecord from the specified memory buffer.
◆	NFRecordGetCbeffProductType (see page 532)	Retrieves the Cbeff product type of the NFRecord.
◆	NFRecordGetCbeffProductTypeMem (see page 532)	Retrieves the Cbeff product type of the packed NFRecord.
◆	NFRecordGetCore (see page 533)	Retrieves the core at the specified index of the NFRecord.
◆	NFRecordGetCoreCapacity (see page 533)	Retrieves the number of cores that the NFRecord can contain.
◆	NFRecordGetCoreCount (see page 534)	Retrieves the number of cores in the NFRecord.
◆	NFRecordGetCores (see page 535)	Copies all cores of NFRecord to the specified array.
◆	NFRecordGetDelta (see page 535)	Retrieves the delta at the specified index of the NFRecord.
◆	NFRecordGetDeltaCapacity (see page 536)	Retrieves the number of deltas that the NFRecord can contain.
◆	NFRecordGetDeltaCount (see page 536)	Retrieves the number of deltas in the NFRecord.
◆	NFRecordGetDeltas (see page 537)	Copies all deltas of NFRecord to the specified array.
◆	NFRecordGetDoubleCore (see page 537)	Retrieves the double core at the specified index of the NFRecord.
◆	NFRecordGetDoubleCoreCapacity (see page 538)	Retrieves the number of double cores that the NFRecord can contain.
◆	NFRecordGetDoubleCoreCount (see page 538)	Retrieves the number of double cores in the NFRecord.
◆	NFRecordGetDoubleCores (see page 539)	Copies all double cores of NFRecord to the specified array.
◆	NFRecordGetG (see page 540)	Retrieves the G of the NFRecord.
◆	NFRecordGetGMem (see page 540)	Retrieves the G of the packed NFRecord.
◆	NFRecordGetHeight (see page 541)	Retrieves the height of the image the NFRecord is made from.
◆	NFRecordGetHeightMem (see page 541)	Retrieves the height of the image the packed NFRecord is made from.
◆	NFRecordGetHorzResolution (see page 542)	Retrieves the horizontal resolution in dpi of the image the NFRecord is made from.
◆	NFRecordGetHorzResolutionMem (see page 542)	Retrieves the horizontal resolution of the image the packed NFRecord is made from.
◆	NFRecordGetImpressionType (see page 543)	Retrieves the impression type of the NFRecord.
◆	NFRecordGetImpressionTypeMem (see page 543)	Retrieves the impression type of the packed NFRecord.
◆	NFRecordGetMinutia (see page 544)	Retrieves the minutia at the specified index of the NFRecord.
◆	NFRecordGetMinutiaCapacity (see page 544)	Retrieves the number of minutiae that the NFRecord can contain.

✦	NFRecordGetMinutiaCount (see page 545)	Retrieves the number of minutiae in the NFRecord.
✦	NFRecordGetMinutiae (see page 546)	Copies all minutiae of NFRecord to the specified array.
✦	NFRecordGetMinutiaFormat (see page 546)	Retrieves the format of the minutiae in NFRecord.
✦	NFRecordGetMinutiaNeighbor (see page 547)	Retrieves the minutia neighbor at the specified index of the minutia at the specified index of the NFRecord.
✦	NFRecordGetMinutiaNeighborCount (see page 547)	Retrieves the number of minutia neighbors in the minutia at the specified index of the NFRecord.
✦	NFRecordGetMinutiaNeighbors (see page 548)	Copies all minutia neighbors of the minutia at the specified index of the NFRecord to the specified array.
✦	NFRecordGetPatternClass (see page 548)	Retrieves the pattern class of the NFRecord. This function returns NFPatternClass (see page 575) if NFPatternClass (see page 575) was set using NFRecordSetPatternClass (see page 567). NfcClassify can be used for classification.
✦	NFRecordGetPatternClassMem (see page 549)	Retrieves the pattern class of the packed NFRecord. This function returns NFPatternClass (see page 575) if NFPatternClass (see page 575) was set using NFRecordSetPatternClass (see page 567). NfcClassify can be used for classification.
✦	NFRecordGetPosition (see page 550)	Retrieves the finger position of the NFRecord.
✦	NFRecordGetPositionMem (see page 550)	Retrieves the finger position of the packed NFRecord.
✦	NFRecordGetQuality (see page 551)	Retrieves the quality of the NFRecord.
✦	NFRecordGetQualityMem (see page 551)	Retrieves the quality of the packed NFRecord.
✦	NFRecordGetRidgeCountsType (see page 552)	Retrieves the ridge counts type the NFRecord contains.
✦	NFRecordGetSize (see page 552)	Calculates packed size of the NFRecord.
✦	NFRecordGetVertResolution (see page 553)	Retrieves the vertical resolution of the image the NFRecord is made from.
✦	NFRecordGetVertResolutionMem (see page 553)	Retrieves the vertical resolution of the image the packed NFRecord is made from.
✦	NFRecordGetWidth (see page 554)	Retrieves the width of the image the NFRecord is made from.
✦	NFRecordGetWidthMem (see page 554)	Retrieves the width of the image the packed NFRecord is made from.
✦	NFRecordInfoDispose (see page 555)	Disposes NERRecordInfo structure.
✦	NFRecordInsertCore (see page 555)	Inserts a NFCore (see page 571) into the NFRecord at the specified index.
✦	NFRecordInsertDelta (see page 556)	Inserts a NFDelta (see page 571) into the NFRecord at the specified index.
✦	NFRecordInsertDoubleCore (see page 556)	Inserts a NFDoubleCore (see page 572) into the NFRecord at the specified index.
✦	NFRecordInsertMinutia (see page 557)	Inserts a NFMinutia (see page 573) into the NFRecord at the specified index.
✦	NFRecordRemoveCore (see page 558)	Removes the core at the specified index of the NFRecord.

◆	NFRecordRemoveDelta (see page 558)	Removes the delta at the specified index of the NFRecord.
◆	NFRecordRemoveDoubleCore (see page 559)	Removes the double core at the specified index of the NFRecord.
◆	NFRecordRemoveMinutia (see page 559)	Removes the minutia at the specified index of the NFRecord.
◆	NFRecordSaveToMemory (see page 560)	Packs the NFRecord into the specified memory buffer.
◆	NFRecordSetCbeffProductType (see page 560)	Sets the Cbeff product type.
◆	NFRecordSetCore (see page 561)	Sets a NFCore (see page 571) at the specified index of the NFRecord.
◆	NFRecordSetCoreCapacity (see page 561)	Sets the number of cores that the NFRecord can contain.
◆	NFRecordSetDelta (see page 562)	Sets a NFDelta (see page 571) at the specified index of the NFRecord.
◆	NFRecordSetDeltaCapacity (see page 563)	Sets the number of deltas that the NFRecord can contain.
◆	NFRecordSetDoubleCore (see page 563)	Sets a NFDoubleCore (see page 572) at the specified index of the NFRecord.
◆	NFRecordSetDoubleCoreCapacity (see page 564)	Sets the number of double cores that the NFRecord can contain.
◆	NFRecordSetG (see page 564)	Sets the G of the NFRecord.
◆	NFRecordSetImpressionType (see page 565)	Sets the impression type of the NFRecord.
◆	NFRecordSetMinutia (see page 565)	Sets a NFMinutia (see page 573) at the specified index of the NFRecord.
◆	NFRecordSetMinutiaCapacity (see page 566)	Sets the number of minutiae that the NFRecord can contain.
◆	NFRecordSetMinutiaFormat (see page 566)	Sets the format of the minutiae in NFRecord.
◆	NFRecordSetMinutiaNeighbor (see page 567)	Sets a NFMinutiaNeighbor (see page 574) at the specified index of the minutia at the specified index of the NFRecord.
◆	NFRecordSetPatternClass (see page 567)	Sets the pattern class of the NFRecord.
◆	NFRecordSetPosition (see page 568)	Sets the finger position of the NFRecord.
◆	NFRecordSetQuality (see page 568)	Sets the quality of the NFRecord.
◆	NFRecordSetRidgeCountsType (see page 569)	Sets the ridge counts type the NFRecord contains.
◆	NFRecordSortMinutiae (see page 569)	Sorts minutiae in NFRecord by the specified order.
◆	NFRecordTruncateMinutiae (see page 570)	Truncates minutiae in NFRecord by peeling off the minutiae convex hull while minutia count is greater than the specified maximal count.
◆	NFRecordTruncateMinutiaeByQuality (see page 570)	Truncates minutiae in NFRecord by removing minutiae which NFMinutia.Quality field value is less than the specified threshold while minutia count is greater than the specified maximal count.
	HNRecord (see page 571)	Handle to NFRecord object.

Macros

Name	Description
NFR_MAX_FINGER_CORE_COUNT (see page 579)	The maximum number of cores a NFRecord can contain.

NFR_MAX_FINGER_DELTA_COUNT (see page 579)	The maximum number of deltas a NFRecord can contain.
NFR_MAX_DIMENSION (see page 579)	The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.
NFR_MAX_DOUBLE_CORE_COUNT (see page 579)	The maximum number of double cores a NFRecord can contain.
NFR_MAX_MINUTIA_COUNT (see page 579)	The maximum number of minutiae a NFRecord can contain.
NFR_RESOLUTION (see page 579)	The resolution of minutiae in dpi, cores, deltas and double cores coordinates in a NFRecord.
NFR_SAVE_BLOCKED_ORIENTS (see page 580)	The flag indicating whether blocked orientations should be packed in NFRecord.
NFR_SKIP_BLOCKED_ORIENTS (see page 580)	The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.
NFR_SKIP_CURVATURES (see page 580)	The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.
NFR_SKIP_GS (see page 580)	The flag indicating whether minutiae gs should be skipped while unpacking or packing NFRecord.
NFR_SKIP_QUALITIES (see page 580)	The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.
NFR_SKIP RIDGE_COUNTS (see page 580)	The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.
NFR_SKIP_SINGULAR_POINTS (see page 581)	The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.
NFR_SAVE_V2 (see page 581)	Saves NFRecord using version 2 algorithm.
NFR_SAVE_V3 (see page 581)	Saves NFRecord using version 3 algorithm.
NFR_MAX_CORE_COUNT (see page 581)	The maximum number of cores a NFRecord can contain.
NFR_MAX_DELTA_COUNT (see page 581)	The maximum number of deltas a NFRecord can contain.
NFR_MAX_FINGER_DIMENSION (see page 582)	The maximum number of finger dimension.
NFR_MAX_FINGER_DOUBLE_CORE_COUNT (see page 582)	The maximum number of double cores a NFRecord can contain.
NFR_MAX_FINGER_MINUTIA_COUNT (see page 582)	The maximum number of finger minutiae a NFRecord can contain.
NFR_MAX_PALM_CORE_COUNT (see page 582)	The maximum number of cores in a palmprint a NFRecord can contain.
NFR_MAX_PALM_DELTA_COUNT (see page 582)	The maximum number of deltas in a palmprint a NFRecord can contain.
NFR_MAX_PALM_DIMENSION (see page 582)	The maximum number of a palmprint dimension.
NFR_MAX_PALM_DOUBLE_CORE_COUNT (see page 583)	The maximum number of double cores in a palmprint a NFRecord can contain.
NFR_MAX_PALM_MINUTIA_COUNT (see page 583)	The maximum number of minutiae in a palmprint a NFRecord can contain.

Structs, Records, Enums

	Name	Description
	NFImpressionType_ (see page 572)	Specifies the impression type.
	NFMinutiaFormat_ (see page 574)	Specifies the minutia format. This enumeration allows a bitwise combination of its member values.
	NFMinutiaOrder_ (see page 574)	Specifies minutia order.
	NFMinutiaType_ (see page 575)	Specifies the minutia type.
	NFPatternClass_ (see page 575)	Specifies the pattern class of the fingerprint.

	NFPosition_ (see page 576)	Specifies the finger position.
	NFRidgeCountsType_ (see page 578)	Specifies the type of ridge counts contained in a NFRecord.
	NFCore_ (see page 571)	Represents a core in a NFRecord.
	NFDelta_ (see page 571)	Represents a delta in a NFRecord.
	NFDoubleCore_ (see page 572)	Represents a double core in a NFRecord.
	NFMinutia_ (see page 573)	Represents a minutia in a NFRecord.
	NFMinutiaNeighbor_ (see page 574)	Represents a minutia neighbor in a NFRecord.
	NFRecordInfo_ (see page 578)	Contains information about the NFRecord.
	NFCore (see page 571)	Represents a core in a NFRecord.
	NFDelta (see page 571)	Represents a delta in a NFRecord.
	NFDoubleCore (see page 572)	Represents a double core in a NFRecord.
	NFImpressionType (see page 572)	Specifies the impression type.
	NFMinutia (see page 573)	Represents a minutia in a NFRecord.
	NFMinutiaFormat (see page 574)	Specifies the minutia format. This enumeration allows a bitwise combination of its member values.
	NFMinutiaNeighbor (see page 574)	Represents a minutia neighbor in a NFRecord.
	NFMinutiaOrder (see page 574)	Specifies minutia order.
	NFMinutiaType (see page 575)	Specifies the minutia type.
	NFPatternClass (see page 575)	Specifies the pattern class of the fingerprint.
	NFPosition (see page 576)	Specifies the finger position.
	NFRecordInfo (see page 578)	Contains information about the NFRecord.
	NFRidgeCountsType (see page 578)	Specifies the type of ridge counts contained in a NFRecord.

6.1.8.2.1 Functions

6.1.8.2.1.1 NFRecordAddCore Function

Adds a NFCore ([see page 571](#)) to the end of NFRecord cores.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
const NFCore * pValue	[out] Pointer to the NFCore (see page 571) to add.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

N_E_INVALID_OPERATION (see page 80)	Number of cores in NFRecord is equal to NFR_MAX_CORE_COUNT (see page 581).

Module

NFRecord Module (see page 518)

6.1.8.2.1.2 NFRecordGetMaxSize Function

Retrieves the maximal size of a packed NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type (for fingerprints) or palmprint values.

C++

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

Parameters

Parameters	Description
NInt version	[in] Specifies algorithm version used by this function. Can have these values: 0, 1, 2, 3. The default value is 0 (in this case third version of algorithm will be used).
NBool isPalm	[in] Boolean value indicating the type of the NFRecord. If set to NTrue (see page 133), NFRecord contains palmprint record.
NMinutiaFormat minutiaFormat	[in] The minutia format.
NInt minutiaCount	[in] The number of minutiae.
NFRidgeCountsType ridgeCountsType	[in] The type of ridge counts.
NInt coreCount	[in] The number of cores.
NInt deltaCount	[in] The number of deltas.
NInt doubleCoreCount	[in] The number of double cores.
NInt boWidth	[in] The width of blocked orientations.
NInt boHeight	[in] The height of blocked orientations.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives maximal size of packed NFRecord.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>minutiaFormat</i> is invalid. - or - <i>ridgeCountsType</i> is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>pSize</i> is NULL (see page 133).

N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>minutiaCount</i> is less than zero or greater than or equal to NFR_MAX_MINUTIA_COUNT (see page 579). - or - <i>coreCount</i> is less than zero or greater than or equal to NFR_MAX_CORE_COUNT (see page 581). - or - <i>deltaCount</i> is less than zero or greater than or equal to NFR_MAX_DELTA_COUNT (see page 581). - or - <i>doubleCoreCount</i> is less than zero or greater than or equal to NFR_MAX_DOUBLE_CORE_COUNT (see page 579). - or - <i>boWidth</i> or <i>boHeight</i> is less than zero or greater than or equal to NFR_MAX_BLOCKED_ORIENTS_DIMENSION.

Remarks

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

The function calculates current (3.0) version packed size of NFRecord.

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

Module

NFRecord Module (see page 518)

6.1.8.2.1.3 NFRecordAddDelta Function

Adds a NFDelta (see page 571) to the end of NFRecord deltas.

C++

```
NResult N_API NFRecordAddDelta(HNFRecord hRecord, const NFDelta * pValue);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
const NFDelta * pValue	[out] Pointer to the NFDelta (see page 571) to add.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Number of deltas in NFRecord is equal to NFR_MAX_DELTA_COUNT (see page 581).

Module

NFRecord Module (see page 518)

6.1.8.2.1.4 NFRecordAddDoubleCore Function

Adds a NFDoubleCore (see page 572) to the end of NFRecord double cores.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
const NFDoubleCore * pValue	[out] Pointer to the NFDoubleCore (see page 572) to add.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Number of double cores in NFRecord is equal to NFR_MAX_DOUBLE_CORE_COUNT (see page 579).

Module

NFRecord Module (see page 518)

6.1.8.2.1.5 NFRecordAddMinutia Function

Adds a NFMinutia (see page 573) to the end of NFRecord minutiae.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
const NFMinutia * pValue	[out] Pointer to the NFMinutia (see page 573) to add.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Number of minutiae in NFRecord is equal to NFR_MAX_MINUTIA_COUNT (see page 579).

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.6 NFRecordCheck Function

Checks if format of the packed NFRecord is correct.

C++

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

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.7 NFRecordClearCores Function

Removes all cores from the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hRecord is NULL (see page 133).

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.8 NFRecordClearDeltas Function

Removes all deltas from the NFRecord.

C++

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


Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.9 NFRecordClearDoubleCores Function

Removes all double cores from the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.10 NFRecordClearMinutiae Function

Removes all minutiae from the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.11 NFRecordClone Function

Creates a copy of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
HNFRecord * pHClonedRecord	[out] Pointer to a HNFRecord (see page 571) that receives handle to newly created NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pHClonedRecord</i> is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

Created object must be deleted using NObjectFree (see page 97) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.12 NFRecordCreate Function

Creates an empty NFRecord.

C++

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

Parameters

Parameters	Description
NUShort width	[in] Specifies width of fingerprint or palmprint image.
NUShort height	[in] Specifies height of fingerprint or palmprint image.
NUShort horzResolution	[in] Specifies horizontal resolution in pixels per inch of fingerprint or palmprint image.
NUShort vertResolution	[in] Specifies vertical resolution in pixels per inch of fingerprint or palmprint image.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function. This parameter is reserved, must be zero.
HNFRecord * pHRecord	[out] Pointer to HNFRecord (see page 571) that receives handle to created NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is zero.
N_E_ARGUMENT_NULL (see page 78)	<i>pHRecord</i> is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

Created object must be deleted using NObjectFree (see page 97) function.

The type of NRecord can be checked by viewing NImpressionType (see page 572) value or using NRecordGetImpressionType (see page 543) function.

Notes

One NRecord can contain only one person's fingerprint or palmprint.

Module

NRecord Module (see page 518)

6.1.8.2.1.13 NRecordCreateEx Function

Creates an empty NRecord.

C++

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

Parameters

Parameters	Description
NBool isPalm	[in] Boolean value indicating the type of NRecord. If set to NTrue (see page 133), NRecord contains palmprint record. If set to NFalse (see page 133), NRecord contains fingerprint record.
NUShort width	[in] Specifies width of fingerprint or palmprint image.
NUShort height	[in] Specifies height of fingerprint or palmprint image.
NUShort horzResolution	[in] Specifies horizontal resolution in pixels per inch of fingerprint or palmprint image.
NUShort vertResolution	[in] Specifies vertical resolution in pixels per inch of fingerprint or palmprint image.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function. This parameter is reserved, must be zero.
HNFRecord * pHRecord	[out] Pointer to HNFRecord (see page 571) that receives handle to created NRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is zero.
N_E_ARGUMENT_NULL (see page 78)	<i>pHRecord</i> is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

Created object must be deleted using `NObjectFree` (see page 97) function.

The type of `NFRecord` can be checked by viewing `NFImpressionType` (see page 572) value or using `NFRecordGetImpressionType` (see page 543) function.

Notes

One `NFRecord` can contain only one person's fingerprint or palmprint.

Module

`NFRecord` Module (see page 518)

6.1.8.2.1.14 NFRecordCreateFromMemory Function

Unpacks a `NFRecord` from the specified memory buffer.

C++

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

Parameters

Parameters	Description
<code>const void * pBuffer</code>	[out] Pointer to memory buffer that contains packed <code>NFRecord</code> .
<code>NSizeType bufferSize</code>	[in] Size of memory buffer that contains packed <code>NFRecord</code> .
<code>NUInt flags</code>	[in] Bitwise combination of zero or more flags that controls behavior of the function.
<code>NFRecordInfo * pInfo</code>	[out] For internal use. Must be NULL (see page 133).
<code>HNFRecord * pHRecord</code>	[out] Pointer to <code>HNFRecord</code> (see page 571) that receives handle to newly created <code>NFRecord</code> object.

Returns

If the function succeeds, the return value is `N_OK` (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pHRecord</i> or <i>buffer</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with <code>NFRecord</code> format.

N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

The following flags are supported:

- NFR_SKIP_BLOCKED_ORIENTS (see page 580)
- NFR_SKIP_CURVATURES (see page 580)
- NFR_SKIP_GS (see page 580)
- NFR_SKIP_QUALITIES (see page 580)
- NFR_SKIP RIDGE_COUNTS (see page 580)
- NFR_SKIP_SINGULAR_POINTS (see page 581).

This function supports both NFRecord version 1.0 and 2.0 formats.

Created object must be deleted using NObjectFree (see page 97) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.15 NFRecordGetCbeffProductType Function

Retrieves the Cbeff product type of the NFRecord.

C++

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

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NFRecord object.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives Cbeff product type.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hRecord or pValue is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.16 NFRecordGetCbeffProductTypeMem Function

Retrieves the Cbeff product type of the packed NFRecord.

C++

```
NResult N_API NFRecordGetCbeffProductTypeMem(const void * pBuffer, NSizeType bufferSize, NUShort * pValue);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives Cbeff product type.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.17 NFRecordGetCore Function

Retrieves the core at the specified index of the NFRecord.

C++

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

Parameters

Parameters	Description
HNfRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of core to retrieve.
NFCore * pValue	[out] Pointer to NFCore (see page 571) that receives core.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount (see page 534) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.18 NFRecordGetCoreCapacity Function

Retrieves the number of cores that the NFRecord can contain.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of cores NFRecord can contain.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hRecord or pValue is NULL (see page 133).

Remarks

Core capacity is the number of cores that the NFRecord can store. Core count (see NFRecordGetCoreCount (see page 534) function) is the number of cores that are actually in the NFRecord.

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

Module

NFRecord Module (see page 518)

6.1.8.2.1.19 NFRecordGetCoreCount Function

Retrieves the number of cores in the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of cores.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hRecord or pValue is NULL (see page 133).

Remarks

Core capacity (see NFRecordGetCoreCapacity (see page 533) and NFRecordSetCoreCapacity (see page 561) functions) is the number of cores that the NFRecord can store. Core count is the number of cores that are actually in the NFRecord.

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

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.20 NFRecordGetCores Function

Copies all cores of NFRecord to the specified array.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NFCore * arCores	[out] Pointer to array of NFCore (see page 571) that receives cores.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>arCores</i> is NULL (see page 133).

Remarks

Array *arCores* points to must be large enough to receive all NFRecord cores. See NFRecordGetCoreCount ([see page 534](#)) function.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.21 NFRecordGetDelta Function

Retrieves the delta at the specified index of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of delta to retrieve.
NFDelta * pValue	[out] Pointer to NFDelta (see page 571) that receives delta.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount (see page 536) function.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.22 NFRecordGetDeltaCapacity Function

Retrieves the number of deltas that the NFRecord can contain.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of deltas NFRecord can contain.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Remarks

Delta capacity is the number of deltas that the NFRecord can store. Delta count (see NFRecordGetDeltaCount ([see page 536](#)) function) is the number of deltas that are actually in the NFRecord.

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

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.23 NFRecordGetDeltaCount Function

Retrieves the number of deltas in the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of deltas.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Remarks

Delta capacity (see [NRecordGetDeltaCapacity](#) (see page 536) and [NRecordSetDeltaCapacity](#) (see page 563) functions) is the number of deltas that the NRecord can store. Delta count is the number of deltas that are actually in the NRecord.

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

Module

NRecord Module (see page 518)

6.1.8.2.1.24 NRecordGetDeltas Function

Copies all deltas of NRecord to the specified array.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NRecord object.
NFDelta * arDeltas	[out] Pointer to array of NFDelta (see page 571) that receives deltas.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>arDeltas</i> is NULL (see page 133).

Remarks

Array *arDeltas* points to must be large enough to receive all NRecord deltas. See [NRecordGetDeltaCount](#) (see page 536) function.

Module

NRecord Module (see page 518)

6.1.8.2.1.25 NRecordGetDoubleCore Function

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

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NRecord object.
NInt index	[in] Index of double core to retrieve.
NFDoubleCore * pValue	[out] Pointer to NFDoubleCore (see page 572) that receives double core.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to double core count obtained using <code>NFRecordGetDoubleCoreCount</code> (see page 538) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.26 NFRecordGetDoubleCoreCapacity Function

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

C++

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

Parameters

Parameters	Description
HNFRecord <i>hRecord</i>	[in] Handle to the NFRecord object.
NInt * <i>pValue</i>	[out] Pointer to NInt (see page 117) that receives number of double cores NFRecord can contain.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Remarks

Double core capacity is the number of double cores that the NFRecord can store. Double core count (see `NFRecordGetDoubleCoreCount` (see page 538) function) is the number of double cores that are actually in the NFRecord.

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

Module

NFRecord Module (see page 518)

6.1.8.2.1.27 NFRecordGetDoubleCoreCount Function

Retrieves the number of double cores in the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of double cores.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Remarks

Double core capacity (see NFRecordGetDoubleCoreCapacity (see page 538) and NFRecordSetDoubleCoreCapacity (see page 564) functions) is the number of double cores that the NFRecord can store. Double core count is the number of double cores that are actually in the NFRecord.

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

Module

NFRecord Module (see page 518)

6.1.8.2.1.28 NFRecordGetDoubleCores Function

Copies all double cores of NFRecord to the specified array.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NFDoubleCore * arDoubleCores	[out] Pointer to array of NFDoubleCore (see page 572) that receives double cores.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>arDoubleCores</i> is NULL (see page 133).

Remarks

Array *arDoubleCores* points to must be large enough to receive all NFRecord double cores. See NFRecordGetDoubleCoreCount (see page 538) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.29 NFRecordGetG Function

Retrieves the G of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NByte * pValue	[out] Pointer to NByte (see page 116) that receives G.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Remarks

G is a global fingerprint feature that reflects ridge density. It can have values from 0 to 255, so it occupies one byte. The bigger is value the bigger is ridge density.

Module

NFRecord Module (see page 518)

6.1.8.2.1.30 NFRecordGetGMem Function

Retrieves the G of the packed NFRecord.

C++

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

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.
NByte * pValue	[out] Pointer to NByte that receives G.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Module

NFRecord Module (see page 518)

6.1.8.2.1.31 NFRecordGetHeight Function

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

C++

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

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NFRecord object.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives height of fingerprint image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.32 NFRecordGetHeightMem Function

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

C++

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

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives height of fingerprint image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).

N_E_END_OF_STREAM (see page 79)	bufferSize is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer buffer points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats. Always returns 1 for version 1.0 format.

Module

NFRecord Module (see page 518)

6.1.8.2.1.33 NFRecordGetHorzResolution Function

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

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives horizontal resolution in pixels per inch of fingerprint image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hRecord or pValue is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.34 NFRecordGetHorzResolutionMem Function

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

C++

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

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives horizontal resolution in pixels per inch of fingerprint image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats. Always returns 500 for version 1.0 format.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.35 NFRecordGetImpressionType Function

Retrieves the impression type of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NFImpressionType * pValue	[out] Pointer to NFImpressionType (see page 572) that receives impression type.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.36 NFRecordGetImpressionTypeMem Function

Retrieves the impression type of the packed NFRecord.

C++

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

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.

NFImpressionType * pValue	[out] Pointer to NFImpressionType (see page 572) that receives impression type.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats. Always returns nfitLiveScanPlain for version 1.0 format.

Module

NFRecord Module (see page 518)

6.1.8.2.1.37 NFRecordGetMinutia Function

Retrieves the minutia at the specified index of the NFRecord.

C++

```
NResult N_API NFRecordGetMinutia(HNFRecord hRecord, NInt index, NFMinutia * pValue);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of minutia to retrieve.
NFMinutia * pValue	[out] Pointer to NFMinutia (see page 573) that receives minutia.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount (see page 545) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.38 NFRecordGetMinutiaCapacity Function

Retrieves the number of minutiae that the NFRecord can contain.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of minutiae NFRecord can contain.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Remarks

Minutia capacity is the number of minutiae that the NFRecord can store. Minutia count (see NFRecordGetMinutiaCount ([see page 545](#)) function) is the number of minutiae that are actually in the NFRecord.

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

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.39 NFRecordGetMinutiaCount Function

Retrieves the number of minutiae in the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of minutiae.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Remarks

Minutia capacity (see NFRecordGetMinutiaCapacity ([see page 544](#)) and NFRecordSetMinutiaCapacity ([see page 566](#)) functions) is the number of minutiae that the NFRecord can store. Minutia count is the number of minutiae that are actually in the NFRecord.

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

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

Module

NFRecord Module (see page 518)

6.1.8.2.1.40 NFRecordGetMinutiae Function

Copies all minutiae of NFRecord to the specified array.

C++

```
NResult N_API NFRecordGetMinutiae(HNFRecord hRecord, NMinutia * arMinutiae);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NMinutia * arMinutiae	[out] Pointer to array of NMinutia (see page 573) that receives minutiae.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>arMinutiae</i> is NULL (see page 133).

Remarks

Array *arMinutiae* points to must be large enough to receive all NFRecord minutiae. See NFRecordGetMinutiaCount (see page 545) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.41 NFRecordGetMinutiaFormat Function

Retrieves the format of the minutiae in NFRecord.

C++

```
NResult N_API NFRecordGetMinutiaFormat(HNFRecord hRecord, NMinutiaFormat * pValue);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NMinutiaFormat * pValue	[out] Pointer to NMinutiaFormat (see page 574) that receives format of minutiae in the NFRecord.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.42 NFRecordGetMinutiaNeighbor Function

Retrieves the minutia neighbor at the specified index of the minutia at the specified index of the NFRecord.

C++

```
NResult N_API NFRecordGetMinutiaNeighbor(HNFRecord hRecord, NInt minutiaIndex, NInt index,  
NMinutiaNeighbor * pValue);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt minutiaIndex	[in] The index of minutia.
NInt index	[in] Index of minutia neighbor to retrieve.
NMinutiaNeighbor * pValue	[out] Pointer to NMinutiaNeighbor (see page 574) that receives minutia neighbor.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>minutiaIndex</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount (see page 545) function. - or - <i>index</i> is less than zero or greater than or equal to minutia neighbor count obtained using NFRecordGetMinutiaNeighborCount (see page 547) function.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.43 NFRecordGetMinutiaNeighborCount Function

Retrieves the number of minutia neighbors in the minutia at the specified index of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt minutiaIndex	[in] The index of minutia.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of minutia neighbors.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to minutia count obtained using NRecordGetMinutiaCount (see page 545) function.

Module

NRecord Module (see page 518)

6.1.8.2.1.44 NRecordGetMinutiaNeighbors Function

Copies all minutia neighbors of the minutia at the specified index of the NRecord to the specified array.

C++

```
NResult N_API NRecordGetMinutiaNeighbors(HNRecord hRecord, NInt minutiaIndex, NMinutiaNeighbor * arMinutiaNeighbors);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.
NInt minutiaIndex	[in] The index of minutia.
NMinutiaNeighbor * arMinutiaNeighbors	[out] Pointer to array of NMinutiaNeighbor (see page 574) that receives minutia neighbors.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>arMinutiaNeighbors</i> is NULL (see page 133).

Remarks

Array *arMinutiaNeighbors* points to must be large enough to receive all minutia neighbors. See NRecordGetMinutiaNeighborCount (see page 547) function.

Module

NRecord Module (see page 518)

6.1.8.2.1.45 NRecordGetPatternClass Function

Retrieves the pattern class of the NRecord.

This function returns NPatternClass (see page 575) if NPatternClass (see page 575) was set using NRecordSetPatternClass (see page 567).

NfcClassify can be used for classification.

C++

```
NResult N_API NRecordGetPatternClass(HNRecord hRecord, NPatternClass * pValue);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.

NFPatternClass * pValue	[out] Pointer to NFPatternClass (see page 575) that receives fingerprint pattern class.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	hRecord or pValue is NULL (see page 133).

Remarks

This function supports both NFPatternClass version 1.0 and 2.0 formats. Always returns *nfpcUnknown* for version 1.0 format.

Module

NFPatternClass Module (see page 518)

6.1.8.2.1.46 NFPatternClassMem Function

Retrieves the pattern class of the packed NFPatternClass.

This function returns NFPatternClass (see page 575) if NFPatternClass (see page 575) was set using NFPatternClassSetPatternClass (see page 567).

NfpClassify can be used for classification.

C++

```
NResult N_API NFPatternClassMem(const void * pBuffer, NSizeType bufferSize, NFPatternClass * pValue);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFPatternClass.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFPatternClass.
NFPatternClass * pValue	[out] Pointer to NFPatternClass (see page 575) that receives fingerprint pattern class.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	buffer or pValue is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	bufferSize is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer buffer points to is inconsistent with NFPatternClass format.

Remarks

This function supports both NFPatternClass version 1.0 and 2.0 formats. Always returns *nfpcUnknown* for version 1.0 format.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.47 NFRecordGetPosition Function

Retrieves the finger position of the NFRecord.

C++

```
NResult N_API NFRecordGetPosition(HNFRecord hRecord, NFPosition * pValue);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NFPosition * pValue	[out] Pointer to NFPosition (see page 576) that receives finger position.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.48 NFRecordGetPositionMem Function

Retrieves the finger position of the packed NFRecord.

C++

```
NResult N_API NFRecordGetPositionMem(const void * pBuffer, NSizeType bufferSize, NFPosition * pValue);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.
NFPosition * pValue	[out] Pointer to NFPosition (see page 576) that receives finger position.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.

N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0, 2.0 and 3.0 formats. Always returns *nfpUnknown* for version 1.0 format.

Module

NFRecord Module (see page 518)

6.1.8.2.1.49 NFRecordGetQuality Function

Retrieves the quality of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NByte * pValue	[out] Pointer to NByte (see page 116) that receives fingerprint quality.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.50 NFRecordGetQualityMem Function

Retrieves the quality of the packed NFRecord.

C++

```
NResult N_API NFRecordGetQualityMem(const void * pBuffer, NSizeType bufferSize, NByte * pValue);
```

Parameters

Parameters	Description
const void * pBuffer	[in] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.
NByte * pValue	[out] Pointer to NByte (see page 116) that receives fingerprint quality.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NRecord format.

Remarks

This function supports both NRecord version 1.0, 2.0 and 3.0 formats. Always returns 0 for version 1.0 format.

Module

NRecord Module ([see page 518](#))

6.1.8.2.1.51 NRecordGetRidgeCountsType Function

Retrieves the ridge counts type the NRecord contains.

C++

```
NResult N_API NRecordGetRidgeCountsType(HNRecord hRecord, NRidgeCountsType * pValue);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.
NRidgeCountsType * pValue	[out] Pointer to NRidgeCountsType (see page 578) that receives ridge counts type stored in NRecord.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Module

NRecord Module ([see page 518](#))

6.1.8.2.1.52 NRecordGetSize Function

Calculates packed size of the NRecord.

C++

```
NResult N_API NRecordGetSize(HNRecord hRecord, NUInt flags, NSizeType * pSize);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives size of packed NRecord.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pSize</i> is NULL (see page 133).

Remarks

The function calculates current (3.0) version packed size of NFRecord. For the list of flags that are supported see NFRecordSaveToMemory (see page 560) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.53 NFRecordGetVertResolution Function

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

C++

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

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NFRecord object.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives vertical resolution in pixels per inch of fingerprint image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.54 NFRecordGetVertResolutionMem Function

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

C++

```
NResult N_API NFRecordGetVertResolutionMem(const void * pBuffer, NSizeType bufferSize, NUShort * pValue);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives vertical resolution in pixels per inch of fingerprint image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NRecord format.

Remarks

This function supports both NRecord version 1.0, 2.0 and 3.0 formats. Always returns 500 for version 1.0 format.

Module

NRecord Module (see page 518)

6.1.8.2.1.55 NRecordGetWidth Function

Retrieves the width of the image the NRecord is made from.

C++

```
NResult N_API NRecordGetWidth(HNRecord hRecord, NUShort * pValue);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives width of fingerprint image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).

Module

NRecord Module (see page 518)

6.1.8.2.1.56 NRecordGetWidthMem Function

Retrieves the width of the image the packed NRecord is made from.

C++

```
NResult N_API NRecordGetWidthMem(const void * pBuffer, NSizeType bufferSize, NUShort * pValue);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFRecord.
NUShort * pValue	[out] Pointer to NUShort (see page 120) that receives width of fingerprint image.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0, 2.0 and 3.0 formats. Always returns 1 for version 1.0 format.

Module

NFRecord Module (see page 518)

6.1.8.2.1.57 NFRecordInfoDispose Function

Disposes NERRecordInfo structure.

C++

```
void N_API NFRecordInfoDispose(NFRecordInfo * pInfo);
```

Parameters

Parameters	Description
NFRecordInfo * pInfo	[out] Pointer to NERRecordInfo structure that should be disposed.

Module

NFRecord Module (see page 518)

6.1.8.2.1.58 NFRecordInsertCore Function

Inserts a NFCore (see page 571) into the NFRecord at the specified index.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index at which core is inserted.
const NFCore * pValue	[out] Pointer to the NFCore (see page 571) to insert.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than core count obtained using NRecordGetCoreCount (see page 534) function.
N_E_INVALID_OPERATION (see page 80)	Number of cores in NRecord is equal to NFR_MAX_CORE_COUNT (see page 581).

Module

NRecord Module (see page 518)

6.1.8.2.1.59 NRecordInsertDelta Function

Inserts a NFDelta (see page 571) into the NRecord at the specified index.

C++

```
NResult N_API NRecordInsertDelta(HNRecord hRecord, NInt index, const NFDelta * pValue);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.
NInt index	[in] Index at which delta is inserted.
const NFDelta * pValue	[out] Pointer to the NFDelta (see page 571) to insert.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than delta count obtained using NRecordGetDeltaCount (see page 536) function.
N_E_INVALID_OPERATION (see page 80)	Number of deltas in NRecord is equal to NFR_MAX_DELTA_COUNT (see page 581).

Module

NRecord Module (see page 518)

6.1.8.2.1.60 NRecordInsertDoubleCore Function

Inserts a NFDoubleCore (see page 572) into the NRecord at the specified index.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index at which double core is inserted.
const NFDoubleCore * pValue	[out] Pointer to the NFDoubleCore (see page 572) to insert.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than double core count obtained using NFRecordGetDoubleCoreCount (see page 538) function.
N_E_INVALID_OPERATION (see page 80)	Number of double core in NFRecord is equal to NFR_MAX_DOUBLE_CORE_COUNT (see page 579).

Module

NFRecord Module (see page 518)

6.1.8.2.1.61 NFRecordInsertMinutia Function

Inserts a NFMinutia (see page 573) into the NFRecord at the specified index.

C++

```
NResult N_API NFRecordInsertMinutia(HNFRecord hRecord, NInt index, const NFMinutia * pValue);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index at which minutia is inserted.
const NFMinutia * pValue	[out] Pointer to the NFMinutia (see page 573) to insert.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than minutia count obtained using NFRecordGetMinutiaCount (see page 545) function.

N_E_INVALID_OPERATION (see page 80)	Number of minutia in NFRecord is equal to NFR_MAX_MINUTIA_COUNT (see page 579).

Module

NFRecord Module (see page 518)

6.1.8.2.1.62 NFRecordRemoveCore Function

Removes the core at the specified index of the NFRecord.

C++

```
NResult N_API NFRecordRemoveCore(HNFRecord hRecord, NInt index);
```

Parameters

Parameters	Description
HNfRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of core to remove.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount (see page 534) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.63 NFRecordRemoveDelta Function

Removes the delta at the specified index of the NFRecord.

C++

```
NResult N_API NFRecordRemoveDelta(HNFRecord hRecord, NInt index);
```

Parameters

Parameters	Description
HNfRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of delta to remove.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount (see page 536) function.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.64 NFRecordRemoveDoubleCore Function

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

C++

```
NResult N_API NFRecordRemoveDoubleCore(HNFRecord hRecord, NInt index);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of double core to remove.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to double core count obtained using NFRecordGetDoubleCoreCount (see page 538) function.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.65 NFRecordRemoveMinutia Function

Removes the minutia at the specified index of the NFRecord.

C++

```
NResult N_API NFRecordRemoveMinutia(HNFRecord hRecord, NInt index);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of minutia to remove.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount (see page 545) function.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.66 NFRecordSaveToMemory Function

Packs the NFRecord into the specified memory buffer.

C++

```
NResult N_API NFRecordSaveToMemory(HNFRecord hRecord, void * pBuffer, NSizeType bufferSize,
NUInt flags, NSizeType * pSize);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
void * pBuffer	[out] Pointer to memory buffer to store packed NFRecord. Can be NULL (see page 133).
NSizeType bufferSize	[in] Size of memory buffer to store packed NFRecord.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives size of packed NFRecord.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>buffer</i> is not NULL (see page 133) and <i>bufferSize</i> is less than size required to store packed NFRecord.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pSize</i> is NULL (see page 133). - or - <i>buffer</i> is NULL (see page 133) and <i>bufferSize</i> is not zero.

Remarks

The function packs NFRecord in current (3.0) version format.

If *buffer* is NULL (see page 133) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as NFRecordGetSize (see page 552) function.

If *buffer* is not NULL (see page 133), *bufferSize* must not be less than value calculated with NFRecordGetSize (see page 552) function.

Note that blocked orientations are not packed by default. The following flags are supported:

- NFR_SAVE_BLOCKED_ORIENTS (see page 580)
- NFR_SKIP_CURVATURES (see page 580)
- NFR_SKIP_GS (see page 580)
- NFR_SKIP_QUALITIES (see page 580)
- NFR_SKIP_RIDGE_COUNTS (see page 580)
- NFR_SKIP_SINGULAR_POINTS (see page 581).

Module

NFRecord Module (see page 518)

6.1.8.2.1.67 NFRecordSetCbeffProductType Function

Sets the Cbeff product type.

C++

```
NResult N_API NFRecordSetCbeffProductType(HNFRecord hRecord, NUShort value);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NUShort value	[in] Cbeff product type.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.68 NFRecordSetCore Function

Sets a NFCore (see page 571) at the specified index of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of core to set.
const NFCore * pValue	[out] Pointer to the NFCore (see page 571) to set.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount (see page 534) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.69 NFRecordSetCoreCapacity Function

Sets the number of cores that the NFRecord can contain.

C++

```
NResult N_API NFRecordSetCoreCapacity(HNFRecord hRecord, NInt value);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt value	[in] New number of cores NFRecord can contain.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>value</i> is less than core count obtained using NFRecordGetCoreCount (see page 534) function.

Remarks

Core capacity is the number of cores that the NFRecord can store. Core count (see NFRecordGetCoreCount (see page 534) function) is the number of cores that are actually in the NFRecord.

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

Module

NFRecord Module (see page 518)

6.1.8.2.1.70 NFRecordSetDelta Function

Sets a NFDelta (see page 571) at the specified index of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of delta to set.
const NFDelta * pValue	[out] Pointer to the NFDelta (see page 571) to set.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount (see page 536) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.71 NFRecordSetDeltaCapacity Function

Sets the number of deltas that the NFRecord can contain.

C++

```
NResult N_API NFRecordSetDeltaCapacity(HNFRecord hRecord, NInt value);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt value	[in] New number of deltas NFRecord can contain.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>value</i> is less than delta count obtained using NFRecordGetDeltaCount (see page 536) function.

Remarks

Delta capacity is the number of deltas that the NFRecord can store. Delta count (see NFRecordGetDeltaCount (see page 536) function) is the number of deltas that are actually in the NFRecord.

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

Module

NFRecord Module (see page 518)

6.1.8.2.1.72 NFRecordSetDoubleCore Function

Sets a NFDoubleCore (see page 572) at the specified index of the NFRecord.

C++

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

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of double core to set.
const NFDoubleCore * pValue	[out] Pointer to the NFDoubleCore (see page 572) to set.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.

N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to double core count obtained using <code>NFRecordGetDoubleCoreCount</code> (see page 538) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.73 NFRecordSetDoubleCoreCapacity Function

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

C++

```
NResult N_API NFRecordSetDoubleCoreCapacity(HNFRecord hRecord, NInt value);
```

Parameters

Parameters	Description
HNFRecord <i>hRecord</i>	[in] Handle to the NFRecord object.
NInt <i>value</i>	[in] New number of double cores NFRecord can contain.

Returns

If the function succeeds, the return value is `N_OK` (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>value</i> is less than double core count obtained using <code>NFRecordGetDoubleCoreCount</code> (see page 538) function.

Remarks

Double core capacity is the number of double cores that the NFRecord can store. Double core count (see `NFRecordGetDoubleCoreCount` (see page 538) function) is the number of double cores that are actually in the NFRecord.

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

Module

NFRecord Module (see page 518)

6.1.8.2.1.74 NFRecordSetG Function

Sets the G of the NFRecord.

C++

```
NResult N_API NFRecordSetG(HNFRecord hRecord, NByte value);
```

Parameters

Parameters	Description
HNFRecord <i>hRecord</i>	[in] Handle to the NFRecord object.
NByte <i>value</i>	[in] New G value.

Returns

If the function succeeds, the return value is `N_OK` (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Remarks

G is a global fingerprint feature that reflects ridge density. It can have values from 0 to 255, so it occupies one byte. The bigger is value the bigger is ridge density.

Module

NFRecord Module (see page 518)

6.1.8.2.1.75 NFRecordSetImpressionType Function

Sets the impression type of the NFRecord.

C++

```
NResult N_API NFRecordSetImpressionType(HNFRecord hRecord, NFImpressionType value);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NFImpressionType value	[in] New impression type value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>value</i> is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.76 NFRecordSetMinutia Function

Sets a NFMinutia (see page 573) at the specified index of the NFRecord.

C++

```
NResult N_API NFRecordSetMinutia(HNFRecord hRecord, NInt index, const NFMinutia * pValue);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NInt index	[in] Index of minutia to set.
const NFMinutia * pValue	[out] Pointer to the NFMinutia (see page 573) to set.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to minutia count obtained using NRecordGetMinutiaCount (see page 545) function.

Module

NRecord Module (see page 518)

6.1.8.2.1.77 NRecordSetMinutiaCapacity Function

Sets the number of minutiae that the NRecord can contain.

C++

```
NResult N_API NRecordSetMinutiaCapacity(HNRecord hRecord, NInt value);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.
NInt value	[in] New number of minutiae NRecord can contain.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>value</i> is less than minutia count obtained using NRecordGetMinutiaCount (see page 545) function.

Remarks

Minutia capacity is the number of minutiae that the NRecord can store. Minutia count (see NRecordGetMinutiaCount (see page 545) function) is the number of minutiae that are actually in the NRecord.

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

Module

NRecord Module (see page 518)

6.1.8.2.1.78 NRecordSetMinutiaFormat Function

Sets the format of the minutiae in NRecord.

C++

```
NResult N_API NRecordSetMinutiaFormat(HNRecord hRecord, NMinutiaFormat value);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NRecord object.

NFMinutiaFormat value	[in] New minutia format value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>value</i> is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.79 NFRecordSetMinutiaNeighbor Function

Sets a NFMinutiaNeighbor (see page 574) at the specified index of the minutia at the specified index of the NFRecord.

C++

```
NResult N_API NFRecordSetMinutiaNeighbor(HNFRecord hRecord, NInt minutiaIndex, NInt index,
const NFMinutiaNeighbor * pValue);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NFRecord object.
NInt minutiaIndex	[in] The index of minutia.
NInt index	[in] Index of minutia neighbor to set.
const NFMinutiaNeighbor * pValue	[out] Pointer to the NFMinutiaNeighbor (see page 574) to set.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>pValue</i> points to is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>minutiaIndex</i> is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount (see page 545) function. - or - <i>index</i> is less than zero or greater than or equal to minutia neighbor count obtained using NFRecordGetMinutiaNeighborCount (see page 547) function.

Module

NFRecord Module (see page 518)

6.1.8.2.1.80 NFRecordSetPatternClass Function

Sets the pattern class of the NFRecord.

C++

```
NResult N_API NFRecordSetPatternClass(HNFRecord hRecord, NFPatternClass value);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NFPatternClass value	[in] New fingerprint pattern class value.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>value</i> is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.81 NFRecordSetPosition Function

Sets the finger position of the NFRecord.

C++

```
NResult N_API NFRecordSetPosition(HNFRecord hRecord, NFPosition value);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NFPosition value	[in] New finger position value.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>value</i> is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module ([see page 518](#))

6.1.8.2.1.82 NFRecordSetQuality Function

Sets the quality of the NFRecord.

C++

```
NResult N_API NFRecordSetQuality(HNFRecord hRecord, NByte value);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.

NByte value	[in] New fingerprint quality value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.83 NFRecordSetRidgeCountsType Function

Sets the ridge counts type the NFRecord contains.

C++

```
NResult N_API NFRecordSetRidgeCountsType(HNFRecord hRecord, NFRidgeCountsType value);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NFRidgeCountsType value	[in] New ridge counts type value.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>value</i> is invalid.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.84 NFRecordSortMinutiae Function

Sorts minutiae in NFRecord by the specified order.

C++

```
NResult N_API NFRecordSortMinutiae(HNFRecord hRecord, NMinutiaOrder order);
```

Parameters

Parameters	Description
HNFRecord hRecord	[in] Handle to the NFRecord object.
NMinutiaOrder order	[in] Specifies minutia order.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Value <i>order</i> is incorrect.
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).

Module

NFRecord Module (see page 518)

6.1.8.2.1.85 NFRecordTruncateMinutiae Function

Truncates minutiae in NFRecord by peeling off the minutiae convex hull while minutia count is greater than the specified maximal count.

C++

```
NResult N_API NFRecordTruncateMinutiae(HNFRecord hRecord, NInt maxCount);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NFRecord object.
NInt maxCount	[in] Maximal minutia count to be present in the NFRecord after truncation.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>maxCount</i> is less than zero or greater than NFR_MAX_MINUTIA_COUNT (see page 579).

Module

NFRecord Module (see page 518)

6.1.8.2.1.86 NFRecordTruncateMinutiaeByQuality Function

Truncates minutiae in NFRecord by removing minutiae which NFMinutia.Quality field value is less than the specified threshold while minutia count is greater than the specified maximal count.

C++

```
NResult N_API NFRecordTruncateMinutiaeByQuality(HNFRecord hRecord, NByte threshold, NInt maxCount);
```

Parameters

Parameters	Description
HNRecord hRecord	[in] Handle to the NFRecord object.
NByte threshold	[in] Specifies minimal NFMinutia (see page 573). Quality field value of minutiae not to be removed.
NInt maxCount	[in] Maximal minutia count to be present in the NFRecord after truncation.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>maxCount</i> is less than zero or greater than NFR_MAX_MINUTIA_COUNT (see page 579).
N_E_ARGUMENT_NULL (see page 78)	<i>hRecord</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>threshold</i> is incorrect.
N_E_INVALID_OPERATION (see page 80)	Minutia format is not NfMinutiaFormat.nfmfHasQuality.

Remarks

Minutia count after truncation may be greater than *maxCount* if there is no enough minutiae with quality less than *threshold*.

Module

NFRecord Module (see page 518)

6.1.8.2.1.87 HNFRecord

Handle to NFRecord object.

Module

NFRecord Module (see page 518)

6.1.8.2.2 Structs, Records, Enums

6.1.8.2.2.1 NFCore Structure

Represents a core in a NFRecord.

C++

```
typedef struct NFCore_ {
    USHORT X;
    USHORT Y;
    NInt Angle;
} NFCore;
```

Members

Members	Description
USHORT X;	X coordinate of this NFCore.
USHORT Y;	Y coordinate of this NFCore.
NInt Angle;	Angle of this NFCore.

Module

NFRecord Module (see page 518)

6.1.8.2.2.2 NFDelta Structure

Represents a delta in a NFRecord.

C++

```
typedef struct NFDelta_ {
    USHORT X;
    USHORT Y;
```

```

NInt Angle1;
NInt Angle2;
NInt Angle3;
} NFDelta;

```

Members

Members	Description
NUShort X;	X coordinate of this NFDelta.
NUShort Y;	Y coordinate of this NFDelta.
NInt Angle1;	First angle of this NFDelta.
NInt Angle2;	Second angle of this NFDelta.
NInt Angle3;	Third angle of this NFDelta.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.2.3 NFDoubleCore Structure

Represents a double core in a NFRecord.

C++

```

typedef struct NFDoubleCore_ {
    NUShort X;
    NUShort Y;
} NFDoubleCore;

```

Members

Members	Description
NUShort X;	X coordinate of this NFDoubleCore.
NUShort Y;	Y coordinate of this NFDoubleCore.

Module

NFRecord Module ([see page 518](#))

6.1.8.2.2.4 NFImpressionType Enumeration

Specifies the impression type.

C++

```

typedef enum NFImpressionType_ {
    nfitLiveScanPlain = 0,
    nfitLiveScanRolled = 1,
    nfitNonliveScanPlain = 2,
    nfitNonliveScanRolled = 3,
    nfitLatentImpression = 4,
    nfitLatentTracing = 5,
    nfitLatentPhoto = 6,
    nfitLatentLift = 7,
    nfitSwipe = 8,
    nfitLiveScanContactless = 9,
    nfitLiveScanPalm = 10,
    nfitNonliveScanPalm = 11,
    nfitLatentPalmImpression = 12,
    nfitLatentPalmTracing = 13,
    nfitLatentPalmPhoto = 14,
    nfitLatentPalmLift = 15
} NFImpressionType;

```

Members

Members	Description
nfitLiveScanPlain = 0	Live-scanned plain fingerprint.
nfitLiveScanRolled = 1	Live-scanned rolled fingerprint.
nfitNonliveScanPlain = 2	Non-live-scanned (from paper) plain fingerprint.
nfitNonliveScanRolled = 3	Non-live-scanned (from paper) rolled fingerprint.
nfitLatentImpression = 4	Latent impression fingerprint.
nfitLatentTracing = 5	Latent tracing fingerprint.
nfitLatentPhoto = 6	Latent photo fingerprint.
nfitLatentLift = 7	Latent lift fingerprint.
nfitSwipe = 8	Live-scanned fingerprint by sliding the finger across a "swipe" sensor.
nfitLiveScanContactless = 9	Live-scanned fingerprint using contactless device.
nfitLiveScanPalm = 10	Live scanned palm.
nfitNonliveScanPalm = 11	Non-live scanned palm.
nfitLatentPalmImpression = 12	Latent palm impression.
nfitLatentPalmTracing = 13	Latent palm tracing.
nfitLatentPalmPhoto = 14	Latent palm photo.
nfitLatentPalmLift = 15	Latent palm lift.

Module

NFRecord Module (see page 518)

6.1.8.2.2.5 NMinutia Structure

Represents a minutia in a NFRecord.

C++

```
typedef struct NMinutia_ {
    NUShort X;
    NUShort Y;
    NMinutiaType Type;
    NByte Angle;
    NByte Quality;
    NByte Curvature;
    NByte G;
} NMinutia;
```

Members

Members	Description
NUShort X;	X coordinate of this NMinutia.
NUShort Y;	Y coordinate of this NMinutia.
NMinutiaType Type;	Type of this NMinutia.
NByte Angle;	Angle of this NMinutia.
NByte Quality;	Quality of this NMinutia.
NByte Curvature;	Ridge curvature near this NMinutia.
NByte G;	G (ridge density) near this NMinutia.

Module

NFRecord Module (see page 518)

6.1.8.2.2.6 NfMinutiaFormat Enumeration

Specifies the minutia format. This enumeration allows a bitwise combination of its member values.

C++

```
typedef enum NfMinutiaFormat_ {  
    nfmfNone = 0,  
    nfmfHasQuality = 1,  
    nfmfHasCurvature = 2,  
    nfmfHasG = 4  
} NfMinutiaFormat;
```

Members

Members	Description
nfmfNone = 0	Indicates that NfMinutia (see page 573) has neither of these fields: Curvature, G or quality. These fields are not preserved during unpacking/packing of NFRecord.
nfmfHasQuality = 1	Indicates that NfMinutia (see page 573). Quality field contains meaningful value and is preserved during unpacking/packing of NFRecord.
nfmfHasCurvature = 2	Indicates that NfMinutia (see page 573). Curvature field contains meaningful value and is preserved during unpacking/packing of NFRecord.
nfmfHasG = 4	Indicates that NfMinutia (see page 573). G field contains meaningful value and is preserved during unpacking/packing of NFRecord.

Module

NFRecord Module (see page 518)

6.1.8.2.2.7 NfMinutiaNeighbor Structure

Represents a minutia neighbor in a NFRecord.

C++

```
typedef struct NfMinutiaNeighbor_ {  
    NInt Index;  
    NByte RidgeCount;  
} NfMinutiaNeighbor;
```

Members

Members	Description
NInt Index;	Index of neighbor minutia.
NByte RidgeCount;	Ridge count to neighbor minutia.

Module

NFRecord Module (see page 518)

6.1.8.2.2.8 NfMinutiaOrder Enumeration

Specifies minutia order.

C++

```
typedef enum NfMinutiaOrder_ {  
    nfmoAscending = 0x01,  
    nfmoDescending = 0x02,  
    nfmoCartesianXY = 0x04,  
    nfmoCartesianYX = 0x08,  
    nfmoAngle = 0x0C,  
}
```

```

    nfmoPolar = 0x10
} NfMinutiaOrder;

```

Members

Members	Description
nfmoAscending = 0x01	Specifies than minutiae are sorted ascending by the specified order.
nfmoDescending = 0x02	Specifies than minutiae are sorted descending by the specified order.
nfmoCartesianXY = 0x04	Specifies than minutiae are sorted by NfMinutia.X field. If NfMinutia.X field of two minutiae are equal NfMinutia.Y field is compared.
nfmoCartesianYX = 0x08	Specifies than minutiae are sorted by NfMinutia.Y field. If NfMinutia.Y field of two minutiae are equal NfMinutia.X is compared.
nfmoAngle = 0x0C	Specifies than minutiae are sorted by NfMinutia.Angle field.
nfmoPolar = 0x10	Specifies than minutiae are sorted by distance from minutiae center of mass. If distance of two minutiae are equal NfMinutia.Angle field is compared.

Module

NFRecord Module (see page 518)

6.1.8.2.2.9 NfMinutiaType Enumeration

Specifies the minutia type.

C++

```

typedef enum NfMinutiaType_ {
    nfmtUnknown = 0,
    nfmtEnd = 1,
    nfmtBifurcation = 2,
    nfmtOther = 3
} NfMinutiaType;

```

Members

Members	Description
nfmtUnknown = 0	The type of the minutia is unknown.
nfmtEnd = 1	The minutia that is an end of a ridge.
nfmtBifurcation = 2	The minutia that is a bifurcation of a ridge.
nfmtOther = 3	The type of the minutia is other than ridge bifurcation or ridge ending.

Module

NFRecord Module (see page 518)

6.1.8.2.2.10 NFPatternClass Enumeration

Specifies the pattern class of the fingerprint.

C++

```

typedef enum NFPatternClass_ {
    nfpcUnknown = 0,
    nfpcPlainArch = 1,
    nfpcTentedArch = 2,
    nfpcRadialLoop = 3,
    nfpcUlnarLoop = 4,
    nfpcPlainWhorl = 5,
}

```



```

    nfpcCentralPocketLoop = 6,
    nfpcDoubleLoop = 7,
    nfpcAccidentalWhorl = 8,
    nfpcWhorl = 9,
    nfpcRightSlantLoop = 10,
    nfpcLeftSlantLoop = 11,
    nfpcScar = 12,
    nfpcAmputation = 15
} NFPatternClass;

```

Members

Members	Description
nfpcUnknown = 0	Unknown pattern class.
nfpcPlainArch = 1	Plain arch pattern class.
nfpcTentedArch = 2	Tented arch pattern class.
nfpcRadialLoop = 3	Radial loop pattern class.
nfpcUlnarLoop = 4	Ulnar loop pattern class.
nfpcPlainWhorl = 5	Plain whorl pattern class.
nfpcCentralPocketLoop = 6	Central pocket loop pattern class.
nfpcDoubleLoop = 7	Double loop pattern class.
nfpcAccidentalWhorl = 8	Accidental whorl pattern class.
nfpcWhorl = 9	Whorl pattern class.
nfpcRightSlantLoop = 10	Right slant loop pattern class.
nfpcLeftSlantLoop = 11	Left slant loop pattern class.
nfpcScar = 12	Scar. Pattern class is not available.
nfpcAmputation = 15	Amputation. Pattern class is not available.

Module

NFRecord Module (see page 518)

6.1.8.2.2.11 NFPosition Enumeration

Specifies the finger position.

C++

```

typedef enum NFPosition_ {
    nfpUnknown = 0,
    nfpRightThumb = 1,
    nfpRightIndex = 2,
    nfpRightMiddle = 3,
    nfpRightRing = 4,
    nfpRightLittle = 5,
    nfpLeftThumb = 6,
    nfpLeftIndex = 7,
    nfpLeftMiddle = 8,
    nfpLeftRing = 9,
    nfpLeftLittle = 10,
    nfpPlainRightThumb = 11,
    nfpPlainLeftThumb = 12,
    nfpPlainRightFourFingers = 13,
    nfpPlainLeftFourFingers = 14,
    nfpPlainThumbs = 15,
    nfpUnknownPalm = 20,
    nfpRightFullPalm = 21,
    nfpRightWritersPalm = 22,
    nfpLeftFullPalm = 23,
    nfpLeftWritersPalm = 24,
    nfpRightLowerPalm = 25,
    nfpRightUpperPalm = 26,
    nfpLeftLowerPalm = 27,
}

```

```

nfpLeftUpperPalm = 28,
nfpRightOther = 29,
nfpLeftOther = 30,
nfpRightInterdigital = 31,
nfpRightThenar = 32,
nfpRightHypothenar = 33,
nfpLeftInterdigital = 34,
nfpLeftThenar = 35,
nfpLeftHypothenar = 36
} NFPosition;

```

Members

Members	Description
nfpUnknown = 0	Unknown finger.
nfpRightThumb = 1	Thumb of the right hand.
nfpRightIndex = 2	Index finger of the right hand.
nfpRightMiddle = 3	Middle finger of the right hand.
nfpRightRing = 4	Ring finger of the right hand.
nfpRightLittle = 5	Little finger of the right hand.
nfpLeftThumb = 6	Thumb of the left hand.
nfpLeftIndex = 7	Index finger of the left hand.
nfpLeftMiddle = 8	Middle finger of the left hand.
nfpLeftRing = 9	Ring finger of the left hand.
nfpLeftLittle = 10	Little finger of the left hand.
nfpPlainRightThumb = 11	Specifies plain right thumb.
nfpPlainLeftThumb = 12	Specifies left plain thumb.
nfpPlainRightFourFingers = 13	Specifies plain right four fingers.
nfpPlainLeftFourFingers = 14	Specifies left four fingers.
nfpPlainThumbs = 15	Specifies plain thumbs.
nfpUnknownPalm = 20	Specifies unknown palm.
nfpRightFullPalm = 21	Specifies right full palm.
nfpRightWritersPalm = 22	Specifies right writers palm.
nfpLeftFullPalm = 23	Specifies full left palm.
nfpLeftWritersPalm = 24	Specifies left writers palm.
nfpRightLowerPalm = 25	Specifies right lower palm.
nfpRightUpperPalm = 26	Specifies right upper palm.
nfpLeftLowerPalm = 27	Specifies lower palm.
nfpLeftUpperPalm = 28	Specifies left upper palm.
nfpRightOther = 29	Unknown finger of the right arm .
nfpLeftOther = 30	Unknown finger of the left arm .
nfpRightInterdigital = 31	Specifies right interdigital.
nfpRightThenar = 32	Specifies right thenar.
nfpRightHypothenar = 33	Specifies right hypothenar.
nfpLeftInterdigital = 34	Specifies left interdigital.
nfpLeftThenar = 35	Specifies left thenar.
nfpLeftHypothenar = 36	Specifies left hypothenar.

Module

NFRecord Module (see page 518)

6.1.8.2.2.12 NFRecordInfo Structure

Contains information about the NFRecord.

C++

```
typedef struct NFRecordInfo_ {  
    NByte MajorVersion;  
    NByte MinorVersion;  
    UInt Size;  
    UInt HeaderSize;  
} NFRecordInfo;
```

Members

Members	Description
NByte MajorVersion;	Major version of the NFRecord.
NByte MinorVersion;	Minor version of the NFRecord.
UInt Size;	Size of the NFRecord.
UInt HeaderSize;	Size of a header.

Module

NFRecord Module (see page 518)

6.1.8.2.2.13 NFRidgeCountsType Enumeration

Specifies the type of ridge counts contained in a NFRecord.

C++

```
typedef enum NFRidgeCountsType_ {  
    nfrctNone = 0,  
    nfrctFourNeighbors = 1,  
    nfrctEightNeighbors = 2,  
    nfrctFourNeighborsWithIndexes = 5,  
    nfrctEightNeighborsWithIndexes = 6,  
    nfrctUnspecified = 128+4  
} NFRidgeCountsType;
```

Members

Members	Description
nfrctNone = 0	The NFRecord does not contain ridge counts.
nfrctFourNeighbors = 1	The NFRecord contains ridge counts to closest minutia in each of the four sectors of each minutia. First sector starts at minutia angle.
nfrctEightNeighbors = 2	The NFRecord contains ridge counts to closest minutia in each of the eight sectors of each minutia. First sector starts at minutia angle.
nfrctFourNeighborsWithIndexes = 5	The NFRecord contains ridge counts to four neighbors of each minutia.
nfrctEightNeighborsWithIndexes = 6	The NFRecord contains ridge counts to eight neighbors of each minutia.
nfrctUnspecified = 128+4	For internal use.

Module

NFRecord Module (see page 518)

6.1.8.2.3 Macros

6.1.8.2.3.1 NFR_MAX_FINGER_CORE_COUNT Macro

The maximum number of cores a NFRecord can contain.

C++

```
#define NFR_MAX_FINGER_CORE_COUNT 15
```

Module

NFRecord Module ([↗](#) see page 518)

6.1.8.2.3.2 NFR_MAX_FINGER_DELTA_COUNT Macro

The maximum number of deltas a NFRecord can contain.

C++

```
#define NFR_MAX_FINGER_DELTA_COUNT 15
```

Module

NFRecord Module ([↗](#) see page 518)

6.1.8.2.3.3 NFR_MAX_DIMENSION Macro

The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.

C++

```
#define NFR_MAX_DIMENSION NFR_MAX_FINGER_DIMENSION
```

Module

NFRecord Module ([↗](#) see page 518)

6.1.8.2.3.4 NFR_MAX_DOUBLE_CORE_COUNT Macro

The maximum number of double cores a NFRecord can contain.

C++

```
#define NFR_MAX_DOUBLE_CORE_COUNT NFR_MAX_FINGER_DOUBLE_CORE_COUNT
```

Module

NFRecord Module ([↗](#) see page 518)

6.1.8.2.3.5 NFR_MAX_MINUTIA_COUNT Macro

The maximum number of minutiae a NFRecord can contain.

C++

```
#define NFR_MAX_MINUTIA_COUNT NFR_MAX_FINGER_MINUTIA_COUNT
```

Module

NFRecord Module ([↗](#) see page 518)

6.1.8.2.3.6 NFR_RESOLUTION Macro

The resolution of minutiae in dpi, cores, deltas and double cores coordinates in a NFRecord.

C++

```
#define NFR_RESOLUTION 500
```

Module

NFRecord Module (see page 518)

6.1.8.2.3.7 NFR_SAVE_BLOCKED_ORIENTS Macro

The flag indicating whether blocked orientations should be packed in NFRecord.

C++

```
#define NFR_SAVE_BLOCKED_ORIENTS 0x00040000
```

Module

NFRecord Module (see page 518)

6.1.8.2.3.8 NFR_SKIP_BLOCKED_ORIENTS Macro

The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.

C++

```
#define NFR_SKIP_BLOCKED_ORIENTS 0x00040000
```

Module

NFRecord Module (see page 518)

6.1.8.2.3.9 NFR_SKIP_CURVATURES Macro

The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.

C++

```
#define NFR_SKIP_CURVATURES 0x00200000
```

Module

NFRecord Module (see page 518)

6.1.8.2.3.10 NFR_SKIP_GS Macro

The flag indicating whether minutiae gs should be skipped while unpacking or packing NFRecord.

C++

```
#define NFR_SKIP_GS 0x00400000
```

Module

NFRecord Module (see page 518)

6.1.8.2.3.11 NFR_SKIP_QUALITIES Macro

The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.

C++

```
#define NFR_SKIP_QUALITIES 0x00100000
```

Module

NFRecord Module (see page 518)

6.1.8.2.3.12 NFR_SKIP_RIDGE_COUNTS Macro

The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.

C++

```
#define NFR_SKIP RIDGE_COUNTS 0x00010000
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.13 NFR_SKIP_SINGULAR_POINTS Macro

The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.

C++

```
#define NFR_SKIP_SINGULAR_POINTS 0x00020000
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.14 NFR_SAVE_V2 Macro

Saves NFRecord using version 2 algorithm.

C++

```
#define NFR_SAVE_V2 0x20000000
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.15 NFR_SAVE_V3 Macro

Saves NFRecord using version 3 algorithm.

C++

```
#define NFR_SAVE_V3 0x30000000
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.16 NFR_MAX_CORE_COUNT Macro

The maximum number of cores a NFRecord can contain.

C++

```
#define NFR_MAX_CORE_COUNT NFR_MAX_FINGER_CORE_COUNT
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.17 NFR_MAX_DELTA_COUNT Macro

The maximum number of deltas a NFRecord can contain.

C++

```
#define NFR_MAX_DELTA_COUNT NFR_MAX_FINGER_DELTA_COUNT
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.18 NFR_MAX_FINGER_DIMENSION Macro

The maximum number of finger dimension.

C++

```
#define NFR_MAX_FINGER_DIMENSION 2047
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.19 NFR_MAX_FINGER_DOUBLE_CORE_COUNT Macro

The maximum number of double cores a NFRecord can contain.

C++

```
#define NFR_MAX_FINGER_DOUBLE_CORE_COUNT 15
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.20 NFR_MAX_FINGER_MINUTIA_COUNT Macro

The maximum number of finger minutiae a NFRecord can contain.

C++

```
#define NFR_MAX_FINGER_MINUTIA_COUNT 255
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.21 NFR_MAX_PALM_CORE_COUNT Macro

The maximum number of cores in a palmprint a NFRecord can contain.

C++

```
#define NFR_MAX_PALM_CORE_COUNT 255
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.22 NFR_MAX_PALM_DELTA_COUNT Macro

The maximum number of deltas in a palmprint a NFRecord can contain.

C++

```
#define NFR_MAX_PALM_DELTA_COUNT 255
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.23 NFR_MAX_PALM_DIMENSION Macro

The maximum number of a palmprint dimension.

C++

```
#define NFR_MAX_PALM_DIMENSION 16383
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.24 NFR_MAX_PALM_DOUBLE_CORE_COUNT Macro

The maximum number of double cores in a palmprint a NFRecord can contain.

C++

```
#define NFR_MAX_PALM_DOUBLE_CORE_COUNT 255
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.3.25 NFR_MAX_PALM_MINUTIA_COUNT Macro

The maximum number of minutiae in a palmprint a NFRecord can contain.

C++

```
#define NFR_MAX_PALM_MINUTIA_COUNT 65535
```

Module

NFRecord Module ([see page 518](#))

6.1.8.2.4 Files**6.1.8.2.4.1 NFRecord.h**

Header file for the NFRecord. Provides functionality for packing, unpacking and editing Neurotechnology Finger Records (NFRecords).

Enumerations

	Name	Description
	NFImpressionType_ (see page 572)	Specifies the impression type.
	NFMinutiaFormat_ (see page 574)	Specifies the minutia format. This enumeration allows a bitwise combination of its member values.
	NFMinutiaOrder_ (see page 574)	Specifies minutia order.
	NFMinutiaType_ (see page 575)	Specifies the minutia type.
	NFPatternClass_ (see page 575)	Specifies the pattern class of the fingerprint.
	NFPosition_ (see page 576)	Specifies the finger position.
	NFRidgeCountsType_ (see page 578)	Specifies the type of ridge counts contained in a NFRecord.
	NFImpressionType (see page 572)	Specifies the impression type.
	NFMinutiaFormat (see page 574)	Specifies the minutia format. This enumeration allows a bitwise combination of its member values.
	NFMinutiaOrder (see page 574)	Specifies minutia order.
	NFMinutiaType (see page 575)	Specifies the minutia type.
	NFPatternClass (see page 575)	Specifies the pattern class of the fingerprint.
	NFPosition (see page 576)	Specifies the finger position.
	NFRidgeCountsType (see page 578)	Specifies the type of ridge counts contained in a NFRecord.

Functions

	Name	Description
≡	NFRecordAddCore (see page 523)	Adds a NFCore (see page 571) to the end of NFRecord cores.
≡	NFRecordAddDelta (see page 525)	Adds a NFDelta (see page 571) to the end of NFRecord deltas.
≡	NFRecordAddDoubleCore (see page 526)	Adds a NFDoubleCore (see page 572) to the end of NFRecord double cores.
≡	NFRecordAddMinutia (see page 526)	Adds a NFMinutia (see page 573) to the end of NFRecord minutiae.
≡	NFRecordCheck (see page 527)	Checks if format of the packed NFRecord is correct.
≡	NFRecordClearCores (see page 527)	Removes all cores from the NFRecord.
≡	NFRecordClearDeltas (see page 527)	Removes all deltas from the NFRecord.
≡	NFRecordClearDoubleCores (see page 528)	Removes all double cores from the NFRecord.
≡	NFRecordClearMinutiae (see page 528)	Removes all minutiae from the NFRecord.
≡	NFRecordClone (see page 529)	Creates a copy of the NFRecord.
≡	NFRecordCreate (see page 529)	Creates an empty NFRecord.
≡	NFRecordCreateEx (see page 530)	Creates an empty NFRecord.
≡	NFRecordCreateFromMemory (see page 531)	Unpacks a NFRecord from the specified memory buffer.
≡	NFRecordGetCbeffProductType (see page 532)	Retrieves the Cbeff product type of the NFRecord.
≡	NFRecordGetCbeffProductTypeMem (see page 532)	Retrieves the Cbeff product type of the packed NFRecord.
≡	NFRecordGetCore (see page 533)	Retrieves the core at the specified index of the NFRecord.
≡	NFRecordGetCoreCapacity (see page 533)	Retrieves the number of cores that the NFRecord can contain.
≡	NFRecordGetCoreCount (see page 534)	Retrieves the number of cores in the NFRecord.
≡	NFRecordGetCores (see page 535)	Copies all cores of NFRecord to the specified array.
≡	NFRecordGetDelta (see page 535)	Retrieves the delta at the specified index of the NFRecord.
≡	NFRecordGetDeltaCapacity (see page 536)	Retrieves the number of deltas that the NFRecord can contain.
≡	NFRecordGetDeltaCount (see page 536)	Retrieves the number of deltas in the NFRecord.
≡	NFRecordGetDeltas (see page 537)	Copies all deltas of NFRecord to the specified array.
≡	NFRecordGetDoubleCore (see page 537)	Retrieves the double core at the specified index of the NFRecord.
≡	NFRecordGetDoubleCoreCapacity (see page 538)	Retrieves the number of double cores that the NFRecord can contain.
≡	NFRecordGetDoubleCoreCount (see page 538)	Retrieves the number of double cores in the NFRecord.
≡	NFRecordGetDoubleCores (see page 539)	Copies all double cores of NFRecord to the specified array.
≡	NFRecordGetG (see page 540)	Retrieves the G of the NFRecord.

◆	NFRecordGetGMem (see page 540)	Retrieves the G of the packed NFRecord.
◆	NFRecordGetHeight (see page 541)	Retrieves the height of the image the NFRecord is made from.
◆	NFRecordGetHeightMem (see page 541)	Retrieves the height of the image the packed NFRecord is made from.
◆	NFRecordGetHorzResolution (see page 542)	Retrieves the horizontal resolution in dpi of the image the NFRecord is made from.
◆	NFRecordGetHorzResolutionMem (see page 542)	Retrieves the horizontal resolution of the image the packed NFRecord is made from.
◆	NFRecordGetImpressionType (see page 543)	Retrieves the impression type of the NFRecord.
◆	NFRecordGetImpressionTypeMem (see page 543)	Retrieves the impression type of the packed NFRecord.
◆	NFRecordGetMaxSize (see page 524)	Retrieves the maximal size of a packed NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type (for fingerprints) or palmprint values.
◆	NFRecordGetMinutia (see page 544)	Retrieves the minutia at the specified index of the NFRecord.
◆	NFRecordGetMinutiaCapacity (see page 544)	Retrieves the number of minutiae that the NFRecord can contain.
◆	NFRecordGetMinutiaCount (see page 545)	Retrieves the number of minutiae in the NFRecord.
◆	NFRecordGetMinutiae (see page 546)	Copies all minutiae of NFRecord to the specified array.
◆	NFRecordGetMinutiaFormat (see page 546)	Retrieves the format of the minutiae in NFRecord.
◆	NFRecordGetMinutiaNeighbor (see page 547)	Retrieves the minutia neighbor at the specified index of the minutia at the specified index of the NFRecord.
◆	NFRecordGetMinutiaNeighborCount (see page 547)	Retrieves the number of minutia neighbors in the minutia at the specified index of the NFRecord.
◆	NFRecordGetMinutiaNeighbors (see page 548)	Copies all minutia neighbors of the minutia at the specified index of the NFRecord to the specified array.
◆	NFRecordGetPatternClass (see page 548)	Retrieves the pattern class of the NFRecord. This function returns NFPatternClass (see page 575) if NFPatternClass (see page 575) was set using NFRecordSetPatternClass (see page 567). NfcClassify can be used for clasification.
◆	NFRecordGetPatternClassMem (see page 549)	Retrieves the pattern class of the packed NFRecord. This function returns NFPatternClass (see page 575) if NFPatternClass (see page 575) was set using NFRecordSetPatternClass (see page 567). NfcClassify can be used for clasification.
◆	NFRecordGetPosition (see page 550)	Retrieves the finger position of the NFRecord.
◆	NFRecordGetPositionMem (see page 550)	Retrieves the finger position of the packed NFRecord.
◆	NFRecordGetQuality (see page 551)	Retrieves the quality of the NFRecord.
◆	NFRecordGetQualityMem (see page 551)	Retrieves the quality of the packed NFRecord.
◆	NFRecordGetRidgeCountsType (see page 552)	Retrieves the ridge counts type the NFRecord contains.
◆	NFRecordGetSize (see page 552)	Calculates packed size of the NFRecord.

◆	NFRecordGetVertResolution (see page 553)	Retrieves the vertical resolution of the image the NFRecord is made from.
◆	NFRecordGetVertResolutionMem (see page 553)	Retrieves the vertical resolution of the image the packed NFRecord is made from.
◆	NFRecordGetWidth (see page 554)	Retrieves the width of the image the NFRecord is made from.
◆	NFRecordGetWidthMem (see page 554)	Retrieves the width of the image the packed NFRecord is made from.
◆	NFRecordInfoDispose (see page 555)	Disposes NERRecordInfo structure.
◆	NFRecordInsertCore (see page 555)	Inserts a NFCore (see page 571) into the NFRecord at the specified index.
◆	NFRecordInsertDelta (see page 556)	Inserts a NFDelta (see page 571) into the NFRecord at the specified index.
◆	NFRecordInsertDoubleCore (see page 556)	Inserts a NFDoubleCore (see page 572) into the NFRecord at the specified index.
◆	NFRecordInsertMinutia (see page 557)	Inserts a NFMinutia (see page 573) into the NFRecord at the specified index.
◆	NFRecordRemoveCore (see page 558)	Removes the core at the specified index of the NFRecord.
◆	NFRecordRemoveDelta (see page 558)	Removes the delta at the specified index of the NFRecord.
◆	NFRecordRemoveDoubleCore (see page 559)	Removes the double core at the specified index of the NFRecord.
◆	NFRecordRemoveMinutia (see page 559)	Removes the minutia at the specified index of the NFRecord.
◆	NFRecordSaveToMemory (see page 560)	Packs the NFRecord into the specified memory buffer.
◆	NFRecordSetCbeffProductType (see page 560)	Sets the Cbeff product type.
◆	NFRecordSetCore (see page 561)	Sets a NFCore (see page 571) at the specified index of the NFRecord.
◆	NFRecordSetCoreCapacity (see page 561)	Sets the number of cores that the NFRecord can contain.
◆	NFRecordSetDelta (see page 562)	Sets a NFDelta (see page 571) at the specified index of the NFRecord.
◆	NFRecordSetDeltaCapacity (see page 563)	Sets the number of deltas that the NFRecord can contain.
◆	NFRecordSetDoubleCore (see page 563)	Sets a NFDoubleCore (see page 572) at the specified index of the NFRecord.
◆	NFRecordSetDoubleCoreCapacity (see page 564)	Sets the number of double cores that the NFRecord can contain.
◆	NFRecordSetG (see page 564)	Sets the G of the NFRecord.
◆	NFRecordSetImpressionType (see page 565)	Sets the impression type of the NFRecord.
◆	NFRecordSetMinutia (see page 565)	Sets a NFMinutia (see page 573) at the specified index of the NFRecord.
◆	NFRecordSetMinutiaCapacity (see page 566)	Sets the number of minutiae that the NFRecord can contain.
◆	NFRecordSetMinutiaFormat (see page 566)	Sets the format of the minutiae in NFRecord.
◆	NFRecordSetMinutiaNeighbor (see page 567)	Sets a NFMinutiaNeighbor (see page 574) at the specified index of the minutia at the specified index of the NFRecord.
◆	NFRecordSetPatternClass (see page 567)	Sets the pattern class of the NFRecord.

	NFRecordSetPosition (see page 568)	Sets the finger position of the NFRecord.
	NFRecordSetQuality (see page 568)	Sets the quality of the NFRecord.
	NFRecordSetRidgeCountsType (see page 569)	Sets the ridge counts type the NFRecord contains.
	NFRecordSortMinutiae (see page 569)	Sorts minutiae in NFRecord by the specified order.
	NFRecordTruncateMinutiae (see page 570)	Truncates minutiae in NFRecord by peeling off the minutiae convex hull while minutia count is greater than the specified maximal count.
	NFRecordTruncateMinutiaeByQuality (see page 570)	Truncates minutiae in NFRecord by removing minutiae which NFMinutia.Quality field value is less than the specified threshold while minutia count is greater than the specified maximal count.

Macros

Name	Description
NFR_MAX_CORE_COUNT (see page 581)	The maximum number of cores a NFRecord can contain.
NFR_MAX_DELTA_COUNT (see page 581)	The maximum number of deltas a NFRecord can contain.
NFR_MAX_DIMENSION (see page 579)	The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.
NFR_MAX_DOUBLE_CORE_COUNT (see page 579)	The maximum number of double cores a NFRecord can contain.
NFR_MAX_FINGER_CORE_COUNT (see page 579)	The maximum number of cores a NFRecord can contain.
NFR_MAX_FINGER_DELTA_COUNT (see page 579)	The maximum number of deltas a NFRecord can contain.
NFR_MAX_FINGER_DIMENSION (see page 582)	The maximum number of finger dimension.
NFR_MAX_FINGER_DOUBLE_CORE_COUNT (see page 582)	The maximum number of double cores a NFRecord can contain.
NFR_MAX_FINGER_MINUTIA_COUNT (see page 582)	The maximum number of finger minutiae a NFRecord can contain.
NFR_MAX_MINUTIA_COUNT (see page 579)	The maximum number of minutiae a NFRecord can contain.
NFR_MAX_PALM_CORE_COUNT (see page 582)	The maximum number of cores in a palmprint a NFRecord can contain.
NFR_MAX_PALM_DELTA_COUNT (see page 582)	The maximum number of deltas in a palmprint a NFRecord can contain.
NFR_MAX_PALM_DIMENSION (see page 582)	The maximum number of a palmprint dimension.
NFR_MAX_PALM_DOUBLE_CORE_COUNT (see page 583)	The maximum number of double cores in a palmprint a NFRecord can contain.
NFR_MAX_PALM_MINUTIA_COUNT (see page 583)	The maximum number of minutiae in a palmprint a NFRecord can contain.
NFR_RESOLUTION (see page 579)	The resolution of minutiae in dpi, cores, deltas and double cores coordinates in a NFRecord.
NFR_SAVE_BLOCKED_ORIENTS (see page 580)	The flag indicating whether blocked orientations should be packed in NFRecord.
NFR_SAVE_V2 (see page 581)	Saves NFRecord using version 2 algorithm.
NFR_SAVE_V3 (see page 581)	Saves NFRecord using version 3 algorithm.
NFR_SKIP_BLOCKED_ORIENTS (see page 580)	The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.
NFR_SKIP_CURVATURES (see page 580)	The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.

NFR_SKIP_GS (see page 580)	The flag indicating whether minutiae gs should be skipped while unpacking or packing NFRecord.
NFR_SKIP_QUALITIES (see page 580)	The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.
NFR_SKIP_RIDGE_COUNTS (see page 580)	The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.
NFR_SKIP_SINGULAR_POINTS (see page 581)	The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.

Module

NFRecord Module (see page 518)

Structures

	Name	Description
	NFCore_ (see page 571)	Represents a core in a NFRecord.
	NFDelta_ (see page 571)	Represents a delta in a NFRecord.
	NFDoubleCore_ (see page 572)	Represents a double core in a NFRecord.
	NFMinutia_ (see page 573)	Represents a minutia in a NFRecord.
	NFMinutiaNeighbor_ (see page 574)	Represents a minutia neighbor in a NFRecord.
	NFRecordInfo_ (see page 578)	Contains information about the NFRecord.
	NFCore (see page 571)	Represents a core in a NFRecord.
	NFDelta (see page 571)	Represents a delta in a NFRecord.
	NFDoubleCore (see page 572)	Represents a double core in a NFRecord.
	NFMinutia (see page 573)	Represents a minutia in a NFRecord.
	NFMinutiaNeighbor (see page 574)	Represents a minutia neighbor in a NFRecord.
	NFRecordInfo (see page 578)	Contains information about the NFRecord.

6.1.8.3 NFTemplate Module

Provides functionality for packing, unpacking and editing Neurotechnology Finger and Palm Templates (NFTemplates).

Files

Name	Description
NFTemplate.h (see page 604)	Header file for the NFTemplate. Provides functionality for packing, unpacking and editing Neurotechnology Fingers/Palms Templates (NFTemplates).

Functions

	Name	Description
	NFTemplateAddRecord (see page 589)	Adds an empty fingerprint or palmprint record to the NFTemplate.
	NFTemplateAddRecordCopy (see page 590)	Adds a copy of the finger or palm record to the NFTemplate.
	NFTemplateAddRecordFromMemory (see page 591)	Unpacks a finger record from the specified memory buffer and adds it to the NFTemplate.
	NFTemplateCalculateSize (see page 592)	Calculates the size of a packed NFTemplate containing the specified number of finger or palm records of specified size.
	NFTemplateCheck (see page 592)	Checks if format of the packed NFTemplate is correct.
	NFTemplateClearRecords (see page 593)	Removes all fingerprint or palmprint records from the NFTemplate.

≡	NFTemplateClone (see page 593)	Creates a copy of the NFTemplate.
≡	NFTemplateCreate (see page 594)	Creates an empty NFTemplate.
≡	NFTemplateCreateEx (see page 594)	Creates an empty NFTemplate.
≡	NFTemplateCreateFromMemory (see page 595)	Unpacks a NFTemplate from the specified memory buffer.
≡	NFTemplateGetRecord (see page 596)	Retrieves the finger record at the specified index of the NFTemplate.
≡	NFTemplateGetRecordCapacity (see page 596)	Retrieves the number of finger records that the NFTemplate can contain..
≡	NFTemplateGetRecordCount (see page 597)	Retrieves the number of finger records in the NFTemplate.
≡	NFTemplateGetRecordCountMem (see page 597)	Retrieves the number of finger records in the packed NFTemplate.
≡	NFTemplateGetSize (see page 598)	Calculates packed size of the NFTemplate.
≡	NFTemplateInfoDispose (see page 598)	Disposes NFTemplateInfo structure.
≡	NFTemplateIsPalm (see page 599)	Checks if the NFTemplate contains palm records.
≡	NFTemplateIsPalmMem (see page 599)	Checks if the NFTemplate contains packed palm records.
≡	NFTemplatePack (see page 600)	Packs packed finger or palm records as NFTemplate into the specified memory buffer.
≡	NFTemplateRemoveRecord (see page 601)	Removes the finger or palm record at the specified index of the NFTemplate.
≡	NFTemplateSaveToMemory (see page 601)	Packs the NFTemplate into the specified memory buffer.
≡	NFTemplateSetRecordCapacity (see page 602)	Sets the number of finger or palm records that the NFTemplate can contain.
≡	NFTemplateUnpack (see page 603)	Unpacks packed finger or palm records from the packed NFTemplate.
	HNFTemplate (see page 604)	Handle to NFTemplate object.

Macros

Name	Description
NFT_MAX_RECORD_COUNT (see page 604)	The maximum number of finger or palm records NFTemplate can contain.
NFT_PROCESS_FIRST_RECORD_ONLY (see page 604)	The flag indicating whether only the first finger/palm record should be unpacked or packed while unpacking or packing NFTemplate.

6.1.8.3.1 Functions

6.1.8.3.1.1 NFTemplateAddRecord Function

Adds an empty fingerprint or palmprint record to the NFTemplate.

C++

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

Parameters

Parameters	Description
HNFTemplate hTemplate	[in] Handle to the NFTemplate object.
NUShort width	[in] Specifies width of fingerprint or palmprint image.

NUShort height	[in] Specifies height of fingerprint or palmprint image.
NUShort horzResolution	[in] Specifies horizontal resolution in pixels per inch of fingerprint image.
NUShort vertResolution	[in] Specifies vertical resolution in pixels per inch of fingerprint image.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
HNFRecord * pHRecord	[out] Pointer to HNFRecord (see page 571) that receives handle to created NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>width</i> or <i>height</i> is zero. - or - <i>horzResolution</i> or <i>vertResolution</i> is zero.
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> or <i>pHRecord</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Number of records in NFTemplate is equal to NFT_MAX_RECORD_COUNT (see page 604).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

For the list of flags that are supported see NFRecordCreate (see page 529) function.

Notes

One NFTemplate object **can not contain** NFRecords with both palmprint and fingerprint records. Using NFTemplateIsPalm (see page 599) function you can check the type of the NFTemplate. If the return value is NTrue (see page 133) it means that NTemplate can contain only palmprint records. Otherwise, NTemplate can contain fingerprint records.

Module

NFTemplate Module (see page 588)

6.1.8.3.1.2 NFTemplateAddRecordCopy Function

Adds a copy of the finger or palm record to the NFTemplate.

C++

```
NResult N_API NFTemplateAddRecordCopy(HNFTemplate hTemplate, HNFRecord hSrcRecord,
HNFRecord * pHRecord);
```

Parameters

Parameters	Description
HNFTemplate hTemplate	[in] Handle to the NFTemplate object.
HNFRecord hSrcRecord	[in] Handle to the NFRecord object.
HNFRecord * pHRecord	[out] Pointer to HNFRecord (see page 571) that receives handle to created NFRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> or <i>hSrcRecord</i> or <i>pHRecord</i> is NULL (see page 133).
N_E_INVALID_OPERATION (see page 80)	Number of records in NTemplate is equal to NFT_MAX_RECORD_COUNT (see page 604).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Module

NTemplate Module (see page 588)

6.1.8.3.1.3 NTemplateAddRecordFromMemory Function

Unpacks a finger record from the specified memory buffer and adds it to the NTemplate.

C++

```
NResult N_API NTemplateAddRecordFromMemory(HNFTemplate hTemplate, const void * pBuffer,
NSizeType bufferSize, NUInt flags, HNFRecord * pHRecord);
```

Parameters

Parameters	Description
HNFTemplate hTemplate	[in] Handle to the NTemplate object.
const void * pBuffer	[out] Pointer to memory buffer that contains packed NRecord.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NRecord.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
HNFRecord * pHRecord	[out] Pointer to HNFRecord (see page 571) that receives handle to created NRecord object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> or <i>pHRecord</i> , or <i>buffer</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NRecord format.
N_E_INVALID_OPERATION (see page 80)	Number of records in NTemplate is equal to NFT_MAX_RECORD_COUNT (see page 604).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

For the list of flags that are supported see NRecordCreateFromMemory (see page 531) function.

Module

NTemplate Module (see page 588)

6.1.8.3.1.4 NFTemplateCalculateSize Function

Calculates the size of a packed NFTemplate containing the specified number of finger or palm records of specified size.

C++

```
NResult N_API NFTemplateCalculateSize(NBool isPalm, NInt recordCount, NSizeType *  
arRecordSizes, NSizeType * pSize);
```

Parameters

Parameters	Description
NBool isPalm	[in] Indicates if the record contains palms.
NInt recordCount	[in] The number of records.
NSizeType * arRecordSizes	[out] Pointer to array of NSizeType that contains packed sizes of each record.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives calculated size of packed NFTemplate.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	Any entry of the array <i>arRecordSizes</i> points to is less than minimal finger record size.
N_E_ARGUMENT_NULL (see page 78)	<i>arRecordSizes</i> or <i>pSize</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>recordCount</i> is less than zero or greater than NFT_MAX_RECORD_COUNT (see page 604).

Remarks

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

Use NFRecordGetMaxSize (see page 524) function to calculate packed size of an individual finger record.

Module

NFTemplate Module (see page 588)

6.1.8.3.1.5 NFTemplateCheck Function

Checks if format of the packed NFTemplate is correct.

C++

```
NResult N_API NFTemplateCheck(const void * pBuffer, NSizeType bufferSize);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFTemplate.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFTemplate.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory <i>buffer</i> points to is inconsistent with NTemplate format.

Module

NTemplate Module (see page 588)

6.1.8.3.1.6 NTemplateClearRecords Function

Removes all fingerprint or palmprint records from the NTemplate.

C++

```
NResult N_API NTemplateClearRecords(HNTemplate hTemplate);
```

Parameters

Parameters	Description
HNTemplate hTemplate	[in] Handle to the NTemplate object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> is NULL (see page 133).

Module

NTemplate Module (see page 588)

6.1.8.3.1.7 NTemplateClone Function

Creates a copy of the NTemplate.

C++

```
NResult N_API NTemplateClone(HNTemplate hTemplate, HNTemplate * pHClonedTemplate);
```

Parameters

Parameters	Description
HNTemplate hTemplate	[in] Handle to the NTemplate object.
HNTemplate * pHClonedTemplate	[out] Pointer to HNTemplate (see page 604) that receives handle to created NTemplate object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> or <i>pHClonedTemplate</i> is NULL (see page 133).

N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

Created object must be deleted using NObjectFree ([see page 97](#)) function.

Module

NFTemplate Module ([see page 588](#))

6.1.8.3.1.8 NFTemplateCreate Function

Creates an empty NFTemplate.

C++

```
NResult N_API NFTemplateCreate(HNFTemplate * pHTemplate);
```

Parameters

Parameters	Description
HNFTemplate * pHTemplate	[out] Pointer to HNFTemplate (see page 604) that receives handle to created NFTemplate object.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pHTemplate</i> is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

Created object must be deleted using NObjectFree ([see page 97](#)) function.

Module

NFTemplate Module ([see page 588](#))

6.1.8.3.1.9 NFTemplateCreateEx Function

Creates an empty NFTemplate.

C++

```
NResult N_API NFTemplateCreateEx(NBool isPalm, NUInt flags, HNFTemplate * pHTemplate);
```

Parameters

Parameters	Description
NBool isPalm	[in] Boolean value indicating the type of NFTemplate. If set to NTrue (see page 133), NFTemplate contains palmprint record. If set to NFalse (see page 133), NFTemplate contains fingerprint record.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function. This parameter is reserved, must be zero.
HNFTemplate * pHTemplate	[out] Pointer to HNFTemplate (see page 604) that receives handle to created NFTemplate object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pHTemplate</i> is NULL (see page 133).
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Module

NTemplate Module (see page 588)

6.1.8.3.1.10 NTemplateCreateFromMemory Function

Unpacks a NTemplate from the specified memory buffer.

C++

```
NResult N_API NTemplateCreateFromMemory(const void * pBuffer, NSizeType bufferSize, NUInt flags, NTemplateInfo * pInfo, HNTemplate * pHTemplate);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NTemplate.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NTemplate.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
NTemplateInfo * pInfo	[out] For internal use. Must be NULL (see page 133).
HNTemplate * pHTemplate	[out] Pointer to HNTemplate (see page 604) that receives handle to newly created NTemplate object.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pHTemplate</i> or <i>buffer</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NTemplate format.
N_E_OUT_OF_MEMORY (see page 81)	There was not enough memory.

Remarks

The following flags are supported:

- NFT_PROCESS_FIRST_RECORD_ONLY (see page 604).

Flags supported by NRecordCreateFromMemory (see page 531) function are applied to each finger or palm record contained in the NTemplate. .

Created object must be deleted using NObjectFree (see page 97) function.

Module

NFTemplate Module ([see page 588](#))

6.1.8.3.1.11 NFTemplateGetRecord Function

Retrieves the finger record at the specified index of the NFTemplate.

C++

```
NResult N_API NFTemplateGetRecord(HNFTemplate hTemplate, NInt index, HNFRRecord * pValue);
```

Parameters

Parameters	Description
HNFTemplate hTemplate	[in] Handle to the NFTemplate object.
NInt index	[in] Index of finger record to retrieve.
HNFRRecord * pValue	[out] Pointer to HNFRRecord (see page 571) that receives handle to NFRecord object.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> or <i>pValue</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater than or equal to finger or palm record count obtained using NFTemplateGetRecordCount (see page 597) function.

Module

NFTemplate Module ([see page 588](#))

6.1.8.3.1.12 NFTemplateGetRecordCapacity Function

Retrieves the number of finger records that the NFTemplate can contain..

C++

```
NResult N_API NFTemplateGetRecordCapacity(HNFTemplate hTemplate, NInt * pValue);
```

Parameters

Parameters	Description
HNFTemplate hTemplate	[in] Handle to the NFTemplate object.
NInt * pValue	[out] Pointer to NInt (see page 117) that receives number of finger or palm records NFTemplate can contain.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> or <i>pValue</i> is NULL (see page 133).

Remarks

Finger and palm record capacity is the number of finger records that the NFTemplate can store. Finger and palm record

count (see `NFTemplateGetRecordCount` (see page 597) function) is the number of finger or palm records that are actually in the `NFTemplate`.

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

Module

`NFTemplate` Module (see page 588)

6.1.8.3.1.13 NFTemplateGetRecordCount Function

Retrieves the number of finger records in the `NFTemplate`.

C++

```
NResult N_API NFTemplateGetRecordCount(HNFTemplate hTemplate, NInt * pValue);
```

Parameters

Parameters	Description
<code>HNFTemplate hTemplate</code>	[in] Handle to <code>NFTemplate</code> object.
<code>NInt * pValue</code>	[out] Pointer to <code>NInt</code> (see page 117) that receives number of finger or palm records.

Returns

If the function succeeds, the return value is `N_OK` (see page 82).

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

Error code	Condition
<code>N_E_ARGUMENT_NULL</code> (see page 78)	<i>hTemplate</i> or <i>pValue</i> is <code>NULL</code> (see page 133).

Remarks

Finger or Palm record capacity (see `NFTemplateGetRecordCapacity` (see page 596) and `NFTemplateSetRecordCapacity` (see page 602) functions) is the number of finger or palm records that the `NFTemplate` can store. Finger or Palm record count is the number of finger or palm records that are actually in the `NFTemplate`.

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

Module

`NFTemplate` Module (see page 588)

6.1.8.3.1.14 NFTemplateGetRecordCountMem Function

Retrieves the number of finger records in the packed `NFTemplate`.

C++

```
NResult N_API NFTemplateGetRecordCountMem(const void * pBuffer, NSizeType bufferSize, NInt * pValue);
```

Parameters

Parameters	Description
<code>const void * pBuffer</code>	[out] Pointer to memory buffer that contains packed <code>NFTemplate</code> .
<code>NSizeType bufferSize</code>	[in] Size of memory buffer that contains packed <code>NFTemplate</code> .
<code>NInt * pValue</code>	[out] Pointer to <code>NInt</code> (see page 117) that receives number of finger or palm records.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> or <i>pValue</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NTemplate format.

Module

NTemplate Module (see page 588)

6.1.8.3.1.15 NTemplateGetSize Function

Calculates packed size of the NTemplate.

C++

```
NResult N_API NTemplateGetSize(HNTemplate hTemplate, NUInt flags, NSizeType * pSize);
```

Parameters

Parameters	Description
HNTemplate hTemplate	[in] Handle to NTemplate object.
NUInt flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives calculated size of packed NTemplate.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> or <i>pSize</i> is NULL (see page 133).

Remarks

For the list of flags that are supported see NTemplateSaveToMemory (see page 601) function.

Module

NTemplate Module (see page 588)

6.1.8.3.1.16 NTemplateInfoDispose Function

Disposes NTemplateInfo structure.

C++

```
void N_API NTemplateInfoDispose(NTemplateInfo * pInfo);
```

Parameters

Parameters	Description
NFTemplateInfo * pInfo	[out] Pointer to NERRecordInfo structure that should be disposed.

Module

NFTemplate Module ([see page 588](#))

6.1.8.3.1.17 NFTemplateIsPalm Function

Checks if the NFTemplate contains palm records.

C++

```
NResult N_API NFTemplateIsPalm(HNFTemplate hTemplate, NBool * pValue);
```

Parameters

Parameters	Description
HNFTemplate hTemplate	[in] Handle to NFTemplate object.
NBool * pValue	[out] Boolean value indicating if the NFTemplate contains palm records.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pValue</i> is NULL (see page 133).

Module

NFTemplate Module ([see page 588](#))

6.1.8.3.1.18 NFTemplateIsPalmMem Function

Checks if the NFTemplate contains packed palm records.

C++

```
NResult N_API NFTemplateIsPalmMem(const void * pBuffer, NSizeType bufferSize, NBool * pValue);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NFTemplate.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
NBool * pValue	[out] Boolean value indicating if the NFTemplate contains palm records.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>pValue</i> is NULL (see page 133).

Module

NFTemplate Module ([see page 588](#))

6.1.8.3.1.19 NFTemplatePack Function

Packs packed finger or palm records as NFTemplate into the specified memory buffer.

C++

```
NResult N_API NFTemplatePack(NBool isPalm, NInt recordCount, const void * * arPRecords,
NSizeType * arRecordSizes, void * pBuffer, NSizeType bufferSize, NSizeType * pSize);
```

Parameters

Parameters	Description
NBool isPalm	[in] Boolean value indicating if the NFTemplate contains palm records. If set to NTrue (see page 133) then NFTemplate contains palm records.
NInt recordCount	[in] The number of finger or palm records.
const void * * arPRecords	[out] Pointer to array of void * that contains packed finger or palm records.
NSizeType * arRecordSizes	[out] Pointer to array of NSizeType (see page 119) that contains packed sizes of each finger or palm record.
void * pBuffer	[out] Pointer to memory buffer to store packed NFTemplate.
NSizeType bufferSize	[in] Size of memory buffer to store packed NFTemplate.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives size of packed NFTemplate.

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>bufferSize</i> is less than size required to store packed NFTemplate. - or - Any entry of the array <i>arRecordSizes</i> points to is less than minimal finger record size. - or - Any entry of the array <i>arRecords</i> points to is NULL (see page 133). - or - Any entry of the array <i>arRecordSizes</i> points to is not equal to size stored in memory buffer according entry of the array <i>arRecords</i> points to.
N_E_ARGUMENT_NULL (see page 78)	<i>arRecords</i> or <i>pBuffer</i> or <i>pSize</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>recordCount</i> is less than zero or greater than NFT_MAX_RECORD_COUNT (see page 604).
N_E_FORMAT (see page 80)	Data in memory buffer any entry of the array <i>arRecords</i> points to is inconsistent with NFRecord format.

Remarks

This is a low-level function and can be changed in future version of the library. *bufferSize* must not be less than value calculated with NFTemplateCalculateSize ([see page 592](#)) function with the same parameter values.

Module

NFTemplate Module (see page 588)

6.1.8.3.1.20 NFTemplateRemoveRecord Function

Removes the finger or palm record at the specified index of the NFTemplate.

C++

```
NResult N_API NFTemplateRemoveRecord(HNFTemplate hTemplate, NInt index);
```

Parameters

Parameters	Description
HNFTemplate hTemplate	[in] Handle to the NFTemplate object.
NInt index	[in] Index of finger or palm record to remove.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>index</i> is less than zero or greater or equal than records count.

Module

NFTemplate Module (see page 588)

6.1.8.3.1.21 NFTemplateSaveToMemory Function

Packs the NFTemplate into the specified memory buffer.

C++

```
NResult N_API NFTemplateSaveToMemory(HNFTemplate hTemplate, void * pBuffer, NSizeType bufferSize, NUInt flags, NSizeType * pSize);
```

Parameters

Parameters	Description
HNFTemplate hTemplate	[in] Handle to NFTemplate object.
void * pBuffer	[out] Pointer to memory buffer to store packed NFTemplate. Can be NULL (see page 133).
NSizeType bufferSize	[in] Size of memory buffer to store packed NFTemplate.
NUInt flags	[in] Bitwise combination of zero or more flags that control behavior of the function.
NSizeType * pSize	[out] Pointer to NSizeType (see page 119) that receives size of packed NFTemplate.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT (see page 78)	<i>buffer</i> is not NULL (see page 133) and <i>bufferSize</i> is less than size required to store packed NTemplate.
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> or <i>pSize</i> is NULL (see page 133). - or - <i>buffer</i> is NULL (see page 133) and <i>bufferSize</i> is not zero.

Remarks

If *buffer* is NULL (see page 133) and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as NTemplateGetSize (see page 598) function.

If *buffer* is not NULL (see page 133), *bufferSize* must not be less than value calculated with NTemplateGetSize (see page 598) function.

The following flags are supported:

- NFT_PROCESS_FIRST_RECORD_ONLY (see page 604).

Flags supported by NFRecordSaveToMemory (see page 560) function are applied to each finger record contained in the NTemplate.

Module

NTemplate Module (see page 588)

6.1.8.3.1.22 NTemplateSetRecordCapacity Function

Sets the number of finger or palm records that the NTemplate can contain.

C++

```
NResult N_API NTemplateSetRecordCapacity(HNTemplate hTemplate, NInt value);
```

Parameters

Parameters	Description
HNTemplate hTemplate	[in] Handle to the NTemplate object.
NInt value	[in] New number of finger records NTemplate can contain.

Returns

If the function succeeds, the return value is N_OK (see page 82).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>hTemplate</i> is NULL (see page 133).
N_E_ARGUMENT_OUT_OF_RANGE (see page 78)	<i>value</i> is less than finger record count obtained using NTemplateGetRecordCount (see page 597) function.

Remarks

Finger or palm record capacity is the number of finger or palm records that the NTemplate can store. Finger or palm record count (see NTemplateGetRecordCount (see page 597) function) is the number of finger or palm records that are actually in the NTemplate.

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

Module

NTemplate Module ([see page 588](#))

6.1.8.3.1.23 NTemplateUnpack Function

Unpacks packed finger or palm records from the packed NTemplate.

C++

```
NResult N_API NTemplateUnpack(const void * pBuffer, NSizeType bufferSize, NBool * pIsPalm,
NByte * pMajorVersion, NByte * pMinorVersion, NUInt * pSize, NByte * pHeaderSize, NInt *
pRecordCount, const void * * arPRecords, NSizeType * arRecordSizes);
```

Parameters

Parameters	Description
const void * pBuffer	[out] Pointer to memory buffer that contains packed NTemplate.
NSizeType bufferSize	[in] Size of memory buffer that contains packed NTemplate.
NBool * pIsPalm	[out] Boolean value indicating if the NTemplate contains palm records. If set to NTrue (see page 133) then NTemplate contains palm records.
NByte * pMajorVersion	[out] Pointer to NByte (see page 116) that receives major version of the packed NTemplate. Can be NULL (see page 133).
NByte * pMinorVersion	[out] Pointer to NByte (see page 116) that receives minor version of the packed NTemplate. Can be NULL (see page 133).
NUInt * pSize	[out] Pointer to NUInt (see page 119) that receives size of the packed NTemplate. Can be NULL (see page 133).
NByte * pHeaderSize	[out] Pointer to NByte (see page 116) that receives header size of the packed NTemplate. Can be NULL (see page 133).
NInt * pRecordCount	[out] Pointer to NInt (see page 117) that receives number of finger records contained in the packed NTemplate. Can be NULL (see page 133).
const void * * arPRecords	[out] Pointer to array of void * that receives pointers to packed finger records contained in the packed NTemplate. Can be NULL (see page 133).
NSizeType * arRecordSizes	[out] Pointer to array of NSizeType (see page 119) that receives sizes of packed finger records contained in the packed NTemplate. Can be NULL (see page 133).

Returns

If the function succeeds, the return value is N_OK ([see page 82](#)).

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

Error code	Condition
N_E_ARGUMENT_NULL (see page 78)	<i>buffer</i> is NULL (see page 133).
N_E_END_OF_STREAM (see page 79)	<i>bufferSize</i> is less than expected.
N_E_FORMAT (see page 80)	Data in memory buffer <i>buffer</i> points to is inconsistent with NTemplate format.

Remarks

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

This function should be first called with *arPRecords* and *arRecordSizes* parameters set to NULL (see page 133) to receive finger or palm record count through *pRecordCount* parameter. Then arrays of finger or palm records count length should be allocated and this function should be called with these array passed in *arPRecords* and *arRecordSizes* parameters.

Module

NFTemplate Module (see page 588)

6.1.8.3.1.24 HNFTemplate

Handle to NFTemplate object.

Module

NFTemplate Module (see page 588)

6.1.8.3.2 Macros**6.1.8.3.2.1 NFT_MAX_RECORD_COUNT Macro**

The maximum number of finger or palm records NFTemplate can contain.

C++

```
#define NFT_MAX_RECORD_COUNT 255
```

Module

NFTemplate Module (see page 588)

6.1.8.3.2.2 NFT_PROCESS_FIRST_RECORD_ONLY Macro

The flag indicating whether only the first finger/palm record should be unpacked or packed while unpacking or packing NFTemplate.

C++

```
#define NFT_PROCESS_FIRST_RECORD_ONLY 0x00000100
```

Module

NFTemplate Module (see page 588)

6.1.8.3.3 Files**6.1.8.3.3.1 NFTemplate.h**

Header file for the NFTemplate. Provides functionality for packing, unpacking and editing Neurotechnology Fingers/Palms Templates (NFTemplates).

Functions

	Name	Description
➦	NFTemplateAddRecord (see page 589)	Adds an empty fingerprint or palmprint record to the NFTemplate.
➦	NFTemplateAddRecordCopy (see page 590)	Adds a copy of the finger or palm record to the NFTemplate.

◆	NTemplateAddRecordFromMemory (see page 591)	Unpacks a finger record from the specified memory buffer and adds it to the NTemplate.
◆	NTemplateCalculateSize (see page 592)	Calculates the size of a packed NTemplate containing the specified number of finger or palm records of specified size.
◆	NTemplateCheck (see page 592)	Checks if format of the packed NTemplate is correct.
◆	NTemplateClearRecords (see page 593)	Removes all fingerprint or palmprint records from the NTemplate.
◆	NTemplateClone (see page 593)	Creates a copy of the NTemplate.
◆	NTemplateCreate (see page 594)	Creates an empty NTemplate.
◆	NTemplateCreateEx (see page 594)	Creates an empty NTemplate.
◆	NTemplateCreateFromMemory (see page 595)	Unpacks a NTemplate from the specified memory buffer.
◆	NTemplateGetRecord (see page 596)	Retrieves the finger record at the specified index of the NTemplate.
◆	NTemplateGetRecordCapacity (see page 596)	Retrieves the number of finger records that the NTemplate can contain..
◆	NTemplateGetRecordCount (see page 597)	Retrieves the number of finger records in the NTemplate.
◆	NTemplateGetRecordCountMem (see page 597)	Retrieves the number of finger records in the packed NTemplate.
◆	NTemplateGetSize (see page 598)	Calculates packed size of the NTemplate.
◆	NTemplateInfoDispose (see page 598)	Disposes NTemplateInfo structure.
◆	NTemplateIsPalm (see page 599)	Checks if the NTemplate contains palm records.
◆	NTemplateIsPalmMem (see page 599)	Checks if the NTemplate contains packed palm records.
◆	NTemplatePack (see page 600)	Packs packed finger or palm records as NTemplate into the specified memory buffer.
◆	NTemplateRemoveRecord (see page 601)	Removes the finger or palm record at the specified index of the NTemplate.
◆	NTemplateSaveToMemory (see page 601)	Packs the NTemplate into the specified memory buffer.
◆	NTemplateSetRecordCapacity (see page 602)	Sets the number of finger or palm records that the NTemplate can contain.
◆	NTemplateUnpack (see page 603)	Unpacks packed finger or palm records from the packed NTemplate.

Macros

Name	Description
NFT_MAX_RECORD_COUNT (see page 604)	The maximum number of finger or palm records NTemplate can contain.
NFT_PROCESS_FIRST_RECORD_ONLY (see page 604)	The flag indicating whether only the first finger/palm record should be unpacked or packed while unpacking or packing NTemplate.

Module

NTemplate Module (see page 588)

6.1.8.4 NTemplates Module

Provides functionality for retrieving information about NTemplates library.

Files

Name	Description
NTemplates.h (🔗 see page 606)	Header file for the NTemplates library.

Functions

	Name	Description
🔗	NTemplatesGetInfo (🔗 see page 606)	Retrieves information about the NTemplates library.

6.1.8.4.1 Functions

6.1.8.4.1.1 NTemplatesGetInfo Function

Retrieves information about the NTemplates (🔗 see page 605) library.

C++

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

Parameters

Parameters	Description
NLibraryInfo * pValue	[out] NLibraryInfo object that contains information about the library.

Returns

If the function succeeds the return value is N_OK (🔗 see page 82). Otherwise, an error code is returned.,

Module

NTemplates Module (🔗 see page 605)

6.1.8.4.2 Files

6.1.8.4.2.1 NTemplates.h

Header file for the NTemplates (🔗 see page 605) library.

Functions

	Name	Description
🔗	NTemplatesGetInfo (🔗 see page 606)	Retrieves information about the NTemplates (🔗 see page 605) library.

Module

NTemplates Module (🔗 see page 605)

6.2 .NET Reference

This chapter contains reference of all libraries included in MegaMatcher On Card 2.0 SDK for .NET developers.

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

Namespaces

Name	Description
Neurotec (see page 607)	Contains classes that provide infrastructure for Neurotechnology components.
Neurotec.Biometrics (see page 639)	Classes under this namespace provides methods and properties for working with biometrical data.
Neurotec.Biometrics.Standards (see page 753)	This namespace contains classes of supported biometrical standards.
Neurotec.DeviceManager (see page 814)	Provides functionality for managing devices, like cameras or fingerprint scanners.
Neurotec.Images (see page 824)	Classes under this namespace provides functionality that enable loading, saving and converting images in various formats.
Neurotec.IO (see page 885)	Classes under this namespace provides infrastructure for Neurotechnology components.
Neurotec.Licensing (see page 895)	Provides functionality for getting, releasing licenses.
Neurotec.SmartCards (see page 899)	Contains classes for working with smartcards.
Neurotec.SmartCards.Biometry (see page 1002)	Contains biometric-specific types for working with smartcards.

6.2.1 Neurotec Namespace

Contains classes that provide infrastructure for Neurotechnology components.

Classes

	Name	Description
	NCore (see page 608)	This class supports internal Neurotechnology libraries infrastructure and should not be used directly in your code.
	NDisposable (see page 615)	Provides a method to release allocated resources.
	NeurotecException (see page 616)	The exception that is thrown when unknown error occurred in one of Neurotechnology libraries.
	NeurotecExceptionBase (see page 616)	The exception that is thrown when unknown error occurred in one of Neurotechnology libraries.
	NIndexPair (see page 617)	Represents pair of indexes.
	NInfo (see page 619)	For internal use only.
	NInfoObject (see page 619)	For internal use only.
	NLibraryInfo (see page 619)	Provides definitions of library info class.
	NObject (see page 621)	Provides functionality for retrieving information about specified object.
	NotActivatedException (see page 623)	The exception that is thrown when one of required Neurotechnology libraries is not activated.
	NProcessorInfo (see page 624)	Provides functionality for getting processor information.
	NRange (see page 626)	Represents range.
	NRational (see page 628)	Represents a signed rational number.
	NResult (see page 629)	Provides functionality for unmanaged functions error handling.
	NURational (see page 635)	Represents an unsigned rational number.
	ParameterException (see page 636)	This type of exception appears when trying to perform unsupported operation with the parameter.
	ParameterReadOnlyException (see page 637)	This type of exception appears when trying to change parameter's value that is read only.

Interfaces

	Name	Description
	INeurotecException (see page 637)	The interface that provides information about Neurotec exceptions.

Module

.NET Reference ([see page 606](#))

Structs, Records, Enums

	Name	Description
	NProcessorVendor (see page 638)	Specifies the processor's vendor name.

Types

Name	Description
NLibraryGetInfo (see page 639)	This delegate supports internal Neurotechnology libraries infrastructure and should not be used directly in your code.

6.2.1.1 Classes

6.2.1.1.1 NCore Class

This class supports internal Neurotechnology libraries infrastructure and should not be used directly in your code.

Class Hierarchy**C#**

```
public static class NCore;
```

NCore Fields

	Name	Description
	DllName (see page 609)	NCore (see page 74) library name.

NCore Methods

	Name	Description
	Alloc (see page 609)	Allocates a block of memory.
	CAlloc (see page 609)	Allocates an array in memory with elements initialized to 0.
	Clear (see page 610)	Clears all elements in the specified array.
	Compare (see page 610)	Compares two memory blocks.
	Copy (see page 611)	Copies data from one memory block to other.
	Fill (see page 612)	Sets memory block to a specified value.
	Free (see page 612)	Frees a specified memory block.
	GetInfo (see page 612)	Retrieves an information about NCore (see page 74) library.
	Move (see page 613)	Moves one memory block to another.
	PtrToArray (see page 613)	Allocates a memory block for an array specified by pointer.
	PtrToStructureArray (see page 614)	Allocates a memory block for a array structure specified by pointer.
	ReAlloc (see page 614)	Reallocate memory blocks.
	WriteBufferToStream (see page 615)	Writes memory buffer to stream.

6.2.1.1.1.1 NCore Fields

6.2.1.1.1.1.1 NCore.DllName Field

NCore (see page 74) library name.

```
C#
public const string DllName = "NCore";
```

6.2.1.1.1.2 NCore Methods

6.2.1.1.1.2.1 Alloc Method

6.2.1.1.1.2.1.1 NCore.Alloc Method (uint)

Allocates a block of memory.

```
C#
public static IntPtr Alloc(uint size);
```

Parameters

Parameters	Description
uint size	Size in bytes of memory to be allocated.

Returns

A representation of a pointer (IntPtr type) to the allocated memory block.

6.2.1.1.1.2.1.2 NCore.Alloc Method (ulong)

Allocates a block of memory.

```
C#
public static IntPtr Alloc(ulong size);
```

Parameters

Parameters	Description
ulong size	Size in bytes of memory to be allocated.

Returns

A representation of a pointer (IntPtr type) to the allocated memory block.

6.2.1.1.1.2.2 CAlloc Method

6.2.1.1.1.2.2.1 NCore.CAlloc Method (uint)

Allocates an array in memory with elements initialized to 0.

```
C#
public static IntPtr CAlloc(uint size);
```

Parameters

Parameters	Description
uint size	Size in bytes of memory to be allocated.

Returns

Pointer to allocated memory block.

6.2.1.1.1.2.2 NCore.CAlloc Method (ulong)

Allocates an array in memory with elements initialized to 0.

C#

```
public static IntPtr CAlloc(ulong size);
```

Parameters

Parameters	Description
ulong size	Size in bytes of memory to be allocated.

Returns

Pointer to allocated memory block.

6.2.1.1.1.2.3 Clear Method

6.2.1.1.1.2.3.1 NCore.Clear Method (IntPtr, uint)

Clears all elements in the specified array.

C#

```
public static void Clear(IntPtr pBlock, uint size);
```

Parameters

Parameters	Description
IntPtr pBlock	A pointer to memory block that contains an array.
uint size	The size in bytes of an array.

6.2.1.1.1.2.3.2 NCore.Clear Method (IntPtr, ulong)

Clears all elements in the specified array.

C#

```
public static void Clear(IntPtr pBlock, ulong size);
```

Parameters

Parameters	Description
IntPtr pBlock	A pointer to memory block that contains an array.
ulong size	The size in bytes of an array.

6.2.1.1.1.2.4 Compare Method

6.2.1.1.1.2.4.1 NCore.Compare Method (IntPtr, IntPtr, uint)

Compares two memory blocks.

C#

```
public static int Compare(IntPtr pBlock1, IntPtr pBlock2, uint size);
```

Parameters

Parameters	Description
IntPtr pBlock1	A pointer to the first memory block.
IntPtr pBlock2	A pointer to the second memory block.
uint size	Number of characters to be compared.

Returns

The value indicating the relationship between the memory blocks.

If return value is less than zero a block specified by pBlock1 is less than a block specified by pBlock2.

If return value is zero a block specified by pBlock1 is identical to a block specified by pBlock2.

If return value is greater than zero a block specified by pBlock1 is greater than a block specified by pBlock2.

6.2.1.1.1.2.4.2 NCore.Compare Method (IntPtr, IntPtr, ulong)

Compares two memory blocks.

C#

```
public static int Compare(IntPtr pBlock1, IntPtr pBlock2, ulong size);
```

Parameters

Parameters	Description
IntPtr pBlock1	A pointer to the first memory block.
IntPtr pBlock2	A pointer to the second memory block.
ulong size	Number of characters to be compared.

Returns

The value indicating the relationship between the memory blocks.

If return value is less than zero a block specified by pBlock1 is less than a block specified by pBlock2.

If return value is zero a block specified by pBlock1 is identical to a block specified by pBlock2.

If return value is greater than zero a block specified by pBlock1 is greater than a block specified by pBlock2.

6.2.1.1.1.2.5 Copy Method**6.2.1.1.1.2.5.1 NCore.Copy Method (IntPtr, IntPtr, uint)**

Copies data from one memory block to other.

C#

```
public static void Copy(IntPtr pDstBlock, IntPtr pSrcBlock, uint size);
```

Parameters

Parameters	Description
IntPtr pDstBlock	A pointer to destination memory block (a memory block where the data from source memory block will be copied).
IntPtr pSrcBlock	A pointer to memory block to be copied to another memory block.
uint size	The size of memory block to be copied (the size is in bytes).

6.2.1.1.1.2.5.2 NCore.Copy Method (IntPtr, IntPtr, ulong)

Copies data from one memory block to other.

C#

```
public static void Copy(IntPtr pDstBlock, IntPtr pSrcBlock, ulong size);
```

Parameters

Parameters	Description
IntPtr pDstBlock	A pointer to destination memory block (a memory block where the data from source memory block will be copied).
IntPtr pSrcBlock	A pointer to memory block to be copied to another memory block.
ulong size	The size of memory block to be copied (the size is in bytes).

6.2.1.1.1.2.6 Fill Method**6.2.1.1.1.2.6.1 NCore.Fill Method (IntPtr, byte, uint)**

Sets memory block to a specified value.

C#

```
public static void Fill(IntPtr pBlock, byte value, uint size);
```

Parameters

Parameters	Description
IntPtr pBlock	A pointer to a destination memory block that contains data.
byte value	A value to set.
uint size	The size in bytes of memory block to be set.

6.2.1.1.1.2.6.2 NCore.Fill Method (IntPtr, byte, ulong)

Sets memory block to a specified value.

C#

```
public static void Fill(IntPtr pBlock, byte value, ulong size);
```

Parameters

Parameters	Description
IntPtr pBlock	A pointer to a destination memory block that contains data.
byte value	A value to set.
ulong size	The size in bytes of memory block to be set.

6.2.1.1.1.2.7 NCore.Free Method

Frees a specified memory block.

C#

```
public static void Free(IntPtr pBlock);
```

Parameters

Parameters	Description
IntPtr pBlock	A pointer to previously allocated memory block to be freed.

Remarks

The Free method frees a memory block that was previously allocated by a call to Alloc (see page 609), CAlloc (see page 609) or ReAlloc (see page 614) methods. The number of freed bytes is equivalent to the number of bytes requested when the block was allocated (or reallocated, in the case of ReAlloc (see page 614)).

6.2.1.1.1.2.8 NCore.GetInfo Method

Retrieves an information about NCore (see page 74) library.

C#

```
public static NLibraryInfo GetInfo();
```

Returns

NLibraryInfo ([see page 619](#)) object.

6.2.1.1.1.2.9 Move Method**6.2.1.1.1.2.9.1 NCore.Move Method (IntPtr, IntPtr, uint)**

Moves one memory block to another.

C#

```
public static void Move(IntPtr pDstBlock, IntPtr pSrcBlock, uint size);
```

Parameters

Parameters	Description
IntPtr pDstBlock	A pointer to destination memory block.
IntPtr pSrcBlock	A pointer to source memory block.
uint size	Size in bytes to be copied.

Remarks

Copies size bytes from pDstBlock to pSrcBlock.

Make sure that the destination buffer is the same size or larger than the source buffer.

6.2.1.1.1.2.9.2 NCore.Move Method (IntPtr, IntPtr, ulong)

Moves one memory block to another.

C#

```
public static void Move(IntPtr pDstBlock, IntPtr pSrcBlock, ulong size);
```

Parameters

Parameters	Description
IntPtr pDstBlock	A pointer to destination memory block.
IntPtr pSrcBlock	A pointer to source memory block.
ulong size	Size in bytes to be copied.

Remarks

Copies size bytes from pDstBlock to pSrcBlock.

Make sure that the destination buffer is the same size or larger than the source buffer.

6.2.1.1.1.2.10 PtrToArray Method**6.2.1.1.1.2.10.1 NCore.PtrToArray Method (IntPtr, int)**

Allocates a memory block for an array specified by pointer.

C#

```
public static byte[] PtrToArray(IntPtr pBuffer, int bufferLength);
```

Parameters

Parameters	Description
IntPtr pBuffer	A pointer to memory block that contains the data to be allocated as an array.
int bufferLength	The length in bytes of memory buffer.

Returns

An array that contains data from memory block.

6.2.1.1.1.2.10.2 NCore.PtrToArray Method (IntPtr, long)

Allocates a memory block for an array specified by pointer.

C#

```
public static byte[] PtrToArray(IntPtr pBuffer, long bufferLength);
```

Parameters

Parameters	Description
IntPtr pBuffer	A pointer to memory block that contains the data to be allocated as an array.
long bufferLength	The length in bytes of memory buffer.

Returns

An array that contains data from memory block.

6.2.1.1.1.2.11 NCore.PtrToStructureArray Method

Allocates a memory block for a array structure specified by pointer.

C#

```
public static Array PtrToStructureArray(IntPtr pValues, uint count, Type type);
```

Parameters

Parameters	Description
IntPtr pValues	A pointer to memory block that contains data.
uint count	Number of items to be allocated as a string data type array.
Type type	Represents type declaration.

Returns

An array that contains data copied from memory block.

6.2.1.1.1.2.12 ReAlloc Method**6.2.1.1.1.2.12.1 NCore.ReAlloc Method (IntPtr, uint)**

Reallocate memory blocks.

C#

```
public static void ReAlloc(ref IntPtr pBlock, uint size);
```

Parameters

Parameters	Description
ref IntPtr pBlock	Pointer to memory block to be allocated.
uint size	Size in bytes of new memory block.

6.2.1.1.1.2.12.2 NCore.ReAlloc Method (IntPtr, ulong)

Reallocate memory blocks.

C#

```
public static void ReAlloc(ref IntPtr pBlock, ulong size);
```

Parameters

Parameters	Description
ref IntPtr pBlock	Pointer to memory block to be allocated.
ulong size	Size in bytes of new memory block.

6.2.1.1.1.2.13 WriteBufferToStream Method**6.2.1.1.1.2.13.1 NCore.WriteBufferToStream Method (IntPtr, int, Stream)**

Writes memory buffer to stream.

C#

```
public static void WriteBufferToStream(IntPtr pBuffer, int bufferLength, Stream stream);
```

Parameters

Parameters	Description
IntPtr pBuffer	A pointer to memory buffer.
int bufferLength	Length of the buffer.
Stream stream	The stream where data should be written.

6.2.1.1.1.2.13.2 NCore.WriteBufferToStream Method (IntPtr, long, Stream)

Writes memory buffer to stream.

C#

```
public static void WriteBufferToStream(IntPtr pBuffer, long bufferLength, Stream stream);
```

Parameters

Parameters	Description
IntPtr pBuffer	A pointer to memory buffer.
long bufferLength	Length of the buffer.
Stream stream	The stream where data should be written.

6.2.1.1.2 NDisposable Class

Provides a method to release allocated resources.

Class Hierarchy

C#

```
public abstract class NDisposable : MarshalByRefObject, IDisposable;
```

NDisposable Methods

	Name	Description
	Dispose ( see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

6.2.1.1.2.1 NDisposable Methods

6.2.1.1.2.1.1 NDisposable.Dispose Method

Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

C#

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

Remarks

This method is used to close or release unmanaged resources. By convention, this method is used for all tasks associated with freeing resources held by an object.

6.2.1.1.3 NeurotecException Class

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

Class Hierarchy

C#

```
public sealed class NeurotecException : NeurotecExceptionBase;
```

INeurotecException Properties

	Name	Description
	Code (see page 638)	Gets an error code of the current exception.
	ManagedStackTrace (see page 638)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 638)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

NeurotecExceptionBase Class

	Name	Description
	Code (see page 617)	Gets an error code of the current exception.
	ManagedStackTrace (see page 617)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	StackTrace (see page 617)	Gets a string representation of the frames on the call stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 617)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

6.2.1.1.4 NeurotecExceptionBase Class

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

Class Hierarchy

C#

```
public abstract class NeurotecExceptionBase : SystemException, INeurotecException;
```

INeurotecException Properties

	Name	Description
	Code (see page 638)	Gets an error code of the current exception.
	ManagedStackTrace (see page 638)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 638)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

NeurotecExceptionBase Class

	Name	Description
	Code (see page 617)	Gets an error code of the current exception.
	ManagedStackTrace (see page 617)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	StackTrace (see page 617)	Gets a string representation of the frames on the call stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 617)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

6.2.1.1.4.1 NeurotecExceptionBase Properties**6.2.1.1.4.1.1 NeurotecExceptionBase.Code Property**

Gets an error code of the current exception.

C#

```
public int Code;
```

Property value

Error code.

6.2.1.1.4.1.2 NeurotecExceptionBase.ManagedStackTrace Property

Gets a string representation of the frames on the managed stack at the time the current exception was thrown.

C#

```
public string ManagedStackTrace;
```

Property value

String that describes the contents of the managed stack.

6.2.1.1.4.1.3 NeurotecExceptionBase.StackTrace Property

Gets a string representation of the frames on the call stack at the time the current exception was thrown.

C#

```
public override string StackTrace;
```

Property value

String that describes the contents of the call stack.

6.2.1.1.4.1.4 NeurotecExceptionBase.UnmanagedStackTrace Property

Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

C#

```
public string UnmanagedStackTrace;
```

Property value

String that describes the contents of the unmanaged stack.

6.2.1.1.5 NIndexPair Structure

Represents pair of indexes.

```
C#
[StructLayout(LayoutKind.Sequential)]
public struct NIndexPair {
}
```

Methods

	Name	Description
	NIndexPair (see page 618)	Initializes a new instance of the NIndexPair structure.

NIndexPair Methods

	Name	Description
	ToString (see page 618)	Retrieves a string representation of the object.

NIndexPair Properties

	Name	Description
	Index1 (see page 618)	Gets or sets first index of this NIndexPair.
	Index2 (see page 618)	Gets or sets second index of this NIndexPair.

6.2.1.1.5.1 NIndexPair.NIndexPair Constructor

Initializes a new instance of the NIndexPair structure.

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

Parameters

Parameters	Description
int index1	First index of this NIndexPair.
int index2	Second index of this NIndexPair.

6.2.1.1.5.2 NIndexPair Methods

6.2.1.1.5.2.1 NIndexPair.ToString Method

Retrieves a string representation of the object.

```
C#
public override string ToString();
```

Returns

A string representation of an object.

6.2.1.1.5.3 NIndexPair Properties

6.2.1.1.5.3.1 NIndexPair.Index1 Property

Gets or sets first index of this NIndexPair (see page 617).

```
C#
public int Index1;
```

6.2.1.1.5.3.2 NIndexPair.Index2 Property

Gets or sets second index of this NIndexPair (see page 617).

```
C#  
  
public int Index2;
```

6.2.1.1.6 NInfo Class

For internal use only.

Class Hierarchy

```
C#  
  
public abstract class NInfo : NDisposable;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

6.2.1.1.7 NInfoObject Class

For internal use only.

Class Hierarchy

```
C#  
  
public abstract class NInfoObject : NDisposable;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

6.2.1.1.8 NLibraryInfo Class

Provides definitions of library info class.

Class Hierarchy

```
C#  
  
public sealed class NLibraryInfo;
```

NLibraryInfo Methods

	Name	Description
	Retrieve (see page 620)	Retrieves information about this library.

NLibraryInfo Properties

	Name	Description
	Activated (see page 620)	Checks if the library is registered.
	Company (see page 620)	Retrieves company name.
	Copyright (see page 620)	Retrieves copyright notice of this library.
	Product (see page 620)	Retrieves product name.
	Title (see page 620)	Retrieves library's title.
	Version (see page 620)	Retrieves library version number.

6.2.1.1.8.1 NLibraryInfo Methods

6.2.1.1.8.1.1 NLibraryInfo.Retrieve Method

Retrieves information about this library.

C#

```
public static NLibraryInfo Retrieve(NLibraryGetInfo getInfo);
```

Parameters

Parameters	Description
NLibraryGetInfo getInfo	NLibraryGetInfo (see page 639) object.

Returns

NLibraryInfo ([see page 619](#)) object which contains information about this library.

6.2.1.1.8.2 NLibraryInfo Properties

6.2.1.1.8.2.1 NLibraryInfo.Activated Property

Checks if the library is registered.

C#

```
public string Activated;
```

6.2.1.1.8.2.2 NLibraryInfo.Company Property

Retrieves company name.

C#

```
public string Company;
```

6.2.1.1.8.2.3 NLibraryInfo.Copyright Property

Retrieves copyright notice of this library.

C#

```
public string Copyright;
```

6.2.1.1.8.2.4 NLibraryInfo.Product Property

Retrieves product name.

C#

```
public string Product;
```

6.2.1.1.8.2.5 NLibraryInfo.Title Property

Retrieves library's title.

C#

```
public string Title;
```

6.2.1.1.8.2.6 NLibraryInfo.Version Property

Retrieves library version number.

C#

```
public Version Version;
```

6.2.1.1.9 NObject Class

Provides functionality for retrieving information about specified object.

Class Hierarchy

C#

```
public abstract class NObject : NDisposable;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NObject Properties

NObject Class

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject.
	Owner (see page 623)	Gets owner of the object.

6.2.1.1.9.1 NObject Methods

6.2.1.1.9.1.1 NObject.CopyParameters Method

Copies parameters values from one NObject ([see page 621](#)) object to another.

C#

```
public static void CopyParameters(NObject srcObj, NObject dstObj);
```

Parameters

Parameters	Description
NObject srcObj	Source NObject (see page 621) object which should be copied.
NObject dstObj	Destination NObject (see page 621) object where source object will be copied.

6.2.1.1.9.1.2 NObject.Free Method

Releases memory resources used by specified object.

C#

```
public static void Free(IntPtr hObject);
```

Parameters

Parameters	Description
IntPtr hObject	Pointer to previously allocated memory block to free.

6.2.1.1.9.1.3 NObject.GetNativeType Method

Retrieves native type of object.

C#

```
public NNativeType GetNativeType();
```

Returns

NativeType value.

6.2.1.1.9.1.4 GetParameter Method**6.2.1.1.9.1.4.1 NObject.GetParameter Method (ushort)**

Gets value of parameter by parameter id.

C#

```
public object GetParameter(ushort parameterId);
```

Parameters

Parameters	Description
ushort parameterId	The parameter identifier.

Returns

The parameter value.

6.2.1.1.9.1.4.2 NObject.GetParameter Method (ushort, ushort)

Gets value of parameter by parameter id.

C#

```
public object GetParameter(ushort partId, ushort parameterId);
```

Parameters

Parameters	Description
ushort partId	Specifies the part of the specified NObject (see page 621).
ushort parameterId	The parameter identifier.

Returns

The parameter value.

6.2.1.1.9.1.5 NObject.Reset Method

Resets all NObject (see page 621) parameters to default values.

C#

```
public void Reset();
```

6.2.1.1.9.1.6 SetParameter Method

6.2.1.1.9.1.6.1 NObject.SetParameter Method (ushort, object)

Sets value of parameter by parameter id.

C#

```
public void SetParameter(ushort parameterId, object value);
```

Parameters

Parameters	Description
ushort parameterId	The parameter identifier.
object value	The parameter value.

6.2.1.1.9.1.6.2 NObject.SetParameter Method (ushort, ushort, object)

Sets value of parameter by parameter id.

C#

```
public void SetParameter(ushort partId, ushort parameterId, object value);
```

Parameters

Parameters	Description
ushort partId	Specifies the part of the specified NObject (see page 621).
ushort parameterId	The parameter identifier.
object value	The parameter value.

6.2.1.1.9.2 NObject Properties

6.2.1.1.9.2.1 NObject.Handle Property

Gets handle to unmanaged NObject ([see page 621](#)).

C#

```
public IntPtr Handle;
```

6.2.1.1.9.2.2 NObject.Owner Property

Gets owner of the object.

C#

```
public NObject Owner;
```

Property value

Object (owner) of NObject ([see page 621](#)) type.

6.2.1.1.10 NotActivatedException Class

The exception that is thrown when one of required Neurotechnology libraries is not activated.

Class Hierarchy

C#

```
public class NotActivatedException : NeurotecExceptionBase;
```

INeurotecException Properties

	Name	Description
	Code (see page 638)	Gets an error code of the current exception.

	ManagedStackTrace (see page 638)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 638)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

NeurotecExceptionBase Class

	Name	Description
	Code (see page 617)	Gets an error code of the current exception.
	ManagedStackTrace (see page 617)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	StackTrace (see page 617)	Gets a string representation of the frames on the call stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 617)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

6.2.1.1.11 NProcessorInfo Class

Provides functionality for getting processor information.

Class Hierarchy

C#

```
public static class NProcessorInfo;
```

NProcessorInfo Methods

	Name	Description
	GetModelInfo (see page 625)	Retrieves processor model information.

NProcessorInfo Properties

	Name	Description
	Is3DNowSupported (see page 625)	Gets a boolean representing if processor supports 3DNow instructions set.
	IsLZCnSupported (see page 625)	Gets a boolean representing if processor supports LZCn instruction.
	IsMmxSupported (see page 625)	Gets a boolean representing if processor supports MMX instructions set.
	IsPopCnSupported (see page 625)	Gets a boolean representing if processor supports PopCn instruction.
	IsSse2Supported (see page 625)	Gets a boolean representing if processor supports SSE2 instructions set.
	IsSse3Supported (see page 625)	Gets a boolean representing if processor supports SSE3 instructions set.
	IsSse41Supported (see page 625)	Gets a boolean representing if processor supports SSE 4.1 instructions set.
	IsSse42Supported (see page 626)	Gets a boolean representing if processor supports SSE 4.2 instructions set.
	IsSse4aSupported (see page 626)	Gets a boolean representing if processor supports SSE 4a instructions set.
	IsSse5Supported (see page 626)	Gets a boolean representing if processor supports SSE5 instructions set.
	IsSseSupported (see page 626)	Gets a boolean representing if processor supports SSE instructions.
	IsSsse3Supported (see page 626)	Gets a boolean representing if processor supports SSSE3 instructions set.
	ModelName (see page 626)	Gets a string representing processor model's name.
	Vendor (see page 626)	Gets current processor's vendor value in vendor's enumeration.
	VendorName (see page 626)	Gets a string representing processor vendor's name.

6.2.1.1.11.1 NProcessorInfo Methods

6.2.1.1.11.1.1 NProcessorInfo.GetModelInfo Method

Retrieves processor model information.

C#

```
public static void GetModelInfo(out int family, out int model, out int stepping);
```

Parameters

Parameters	Description
out int family	Integer representing processor's family.
out int model	Integer representing processor's model name.
out int stepping	Integer representing processor's stepping.

6.2.1.1.11.2 NProcessorInfo Properties

6.2.1.1.11.2.1 NProcessorInfo.Is3DNowSupported Property

Gets a boolean representing if processor supports 3DNow instructions set.

C#

```
public static bool Is3DNowSupported;
```

6.2.1.1.11.2.2 NProcessorInfo.IsLZCntSupported Property

Gets a boolean representing if processor supports LZCnt instruction.

C#

```
public static bool IsLZCntSupported;
```

6.2.1.1.11.2.3 NProcessorInfo.IsMmxSupported Property

Gets a boolean representing if processor supports MMX instructions set.

C#

```
public static bool IsMmxSupported;
```

6.2.1.1.11.2.4 NProcessorInfo.IsPopCntSupported Property

Gets a boolean representing if processor supports PopCnt instruction.

C#

```
public static bool IsPopCntSupported;
```

6.2.1.1.11.2.5 NProcessorInfo.IsSse2Supported Property

Gets a boolean representing if processor supports SSE2 instructions set.

C#

```
public static bool IsSse2Supported;
```

6.2.1.1.11.2.6 NProcessorInfo.IsSse3Supported Property

Gets a boolean representing if processor supports SSE3 instructions set.

C#

```
public static bool IsSse3Supported;
```

6.2.1.1.11.2.7 NProcessorInfo.IsSse41Supported Property

Gets a boolean representing if processor supports SSE 4.1 instructions set.

C#

```
public static bool IsSse41Supported;
```

6.2.1.1.11.2.8 NProcessorInfo.IsSse42Supported Property

Gets a boolean representing if processor supports SSE 4.2 instructions set.

C#

```
public static bool IsSse42Supported;
```

6.2.1.1.11.2.9 NProcessorInfo.IsSse4aSupported Property

Gets a boolean representing if processor supports SSE 4a instructions set.

C#

```
public static bool IsSse4aSupported;
```

6.2.1.1.11.2.10 NProcessorInfo.IsSse5Supported Property

Gets a boolean representing if processor supports SSE5 instructions set.

C#

```
public static bool IsSse5Supported;
```

6.2.1.1.11.2.11 NProcessorInfo.IsSseSupported Property

Gets a boolean representing if processor supports SSE instructions.

C#

```
public static bool IsSseSupported;
```

6.2.1.1.11.2.12 NProcessorInfo.IsSsse3Supported Property

Gets a boolean representing if processor supports SSSE3 instructions set.

C#

```
public static bool IsSsse3Supported;
```

6.2.1.1.11.2.13 NProcessorInfo.ModelName Property

Gets a string representing processor model's name.

C#

```
public static string ModelName;
```

6.2.1.1.11.2.14 NProcessorInfo.Vendor Property

Gets current processor's vendor value in vendor's enumeration.

C#

```
public static NProcessorVendor Vendor;
```

6.2.1.1.11.2.15 NProcessorInfo.VendorName Property

Gets a string representing processor vendor's name.

C#

```
public static string VendorName;
```

6.2.1.1.12 NRange Structure

Represents range.

```
C#  
[StructLayout(LayoutKind.Sequential)]  
public struct NRange {  
}
```

Methods

	Name	Description
	NRange (see page 627)	Initializes a new instance of the NRange structure.

NRange Methods

	Name	Description
	ToString (see page 627)	Retrieves a string representation of the range object.

NRange Properties

	Name	Description
	From (see page 627)	Gets or sets lower bound of a range.
	To (see page 627)	Gets or sets upper bound of a range.

6.2.1.1.12.1 NRange.NRange Constructor

Initializes a new instance of the NRange structure.

```
C#  
public NRange(int from, int to);
```

Parameters

Parameters	Description
int from	lower bound of a range to be created.
int to	upper bound of a range to be created.

6.2.1.1.12.2 NRange Methods

6.2.1.1.12.2.1 NRange.ToString Method

Retrieves a string representation of the range object.

```
C#  
public override string ToString();
```

Returns

String representation of the range.

6.2.1.1.12.3 NRange Properties

6.2.1.1.12.3.1 NRange.From Property

Gets or sets lower bound of a range.

```
C#  
public int From;
```

6.2.1.1.12.3.2 NRange.To Property

Gets or sets upper bound of a range.

C#

```
public int To;
```

6.2.1.1.13 NRational Structure

Represents a signed rational number.

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct NRational {
    public static readonly NRational Empty = new NRational(0, 0);
}
```

NRational Fields

	Name	Description
	Empty (see page 628)	Represents that NRational is a null reference.

Methods

	Name	Description
	NRational (see page 628)	Initializes a new instance of the NRational structure.

NRational Methods

	Name	Description
	ToString (see page 629)	Retrieves a string representation of the object.

NRational Properties

	Name	Description
	Denominator (see page 629)	Sets or retrieves the NRational value Denominator.
	Numerator (see page 629)	Sets or retrieves the NRational value Numerator.

6.2.1.1.13.1 NRational.NRational Constructor

Initializes a new instance of the NRational structure.

C#

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

Parameters

Parameters	Description
int numerator	Numerator (see page 629) of this NRational.
int denominator	Denominator (see page 629) of this NRational.

6.2.1.1.13.2 NRational Fields

6.2.1.1.13.2.1 NRational.Empty Field

Represents that NRational [\(see page 628\)](#) is a null reference.

C#

```
public static readonly NRational Empty = new NRational(0, 0);
```

6.2.1.1.13.3 NRational Methods

6.2.1.1.13.3.1 NRational.ToString Method

Retrieves a string representation of the object.

C#

```
public override string ToString();
```

Returns

A string representation of an object.

6.2.1.1.13.4 NRational Properties

6.2.1.1.13.4.1 NRational.Denominator Property

Sets or retrieves the NRational ([see page 628](#)) value Denominator.

C#

```
public int Denominator;
```

6.2.1.1.13.4.2 NRational.Numerator Property

Sets or retrieves the NRational ([see page 628](#)) value Numerator.

C#

```
public int Numerator;
```

6.2.1.1.14 NResult Class

Provides functionality for unmanaged functions error handling.

Class Hierarchy

C#

```
public static class NResult;
```

NResult Fields

	Name	Description
	EArgument (see page 630)	Invalid argument.
	EArgumentNull (see page 630)	Argument is NULL (see page 133).
	EArgumentOutOfRange (see page 630)	Argument is out of range.
	EArithmetic (see page 631)	Arithmetic error occurred.
	EClr (see page 631)	CLR error occurred.
	ECom (see page 631)	COM error occurred.
	ECore (see page 631)	Core error occurred.
	EDirectoryNotFound (see page 631)	Directory not found.
	EDriveNotFound (see page 631)	Drive not found.
	EEndOfStream (see page 631)	Unexpected end of stream.
	EExternal (see page 631)	External error occurred.
	EFailed (see page 631)	Operation failed.
	EFileLoad (see page 632)	Error loading module from file.
	EFileNotFound (see page 632)	File not found.
	EFormat (see page 632)	Argument is of invalid format.

	EIndexOutOfRange (see page 632)	Access with index was out of range.
	EInvalidCast (see page 632)	The cast is invalid.
	EInvalidEnumArgument (see page 632)	Argument is an invalid enum value.
	EInvalidOperation (see page 632)	Attempted to execute operation that is not valid for the object.
	EIO (see page 632)	IO (see page 885) error occurred.
	ENotActivated (see page 632)	Operation is not activated.
	ENotImplemented (see page 633)	Operation is not implemented.
	ENotSupported (see page 633)	Operation is not supported.
	ENullReference (see page 633)	Attempted to reference a NULL (see page 133) object.
	EOutOfMemory (see page 633)	Out of memory.
	EOverflow (see page 633)	Overflow occurred.
	EParameter (see page 633)	Invalid parameter Id.
	EParameterReadOnly (see page 633)	Attempted to set read-only parameter.
	EPathTooLong (see page 633)	Path is too long.
	ESecurity (see page 633)	Security error occurred.
	ESys (see page 634)	Sys error occurred.
	EWin32 (see page 634)	Win32 error occurred.
	Ok (see page 634)	No error.

NResult Methods

	Name	Description
	Check (see page 634)	Checks if the result of a function indicates that an error has occurred and if so throws an exception.
	IsFailed (see page 634)	Checks if the result of a function indicates that an error has occurred.
	IsSucceeded (see page 634)	Checks if the result of a function indicates that no error has occurred.
	RaiseError (see page 635)	Creates and throws exception object which represents the error.
	SetError (see page 635)	For internal use only.

6.2.1.1.14.1 NResult Fields

6.2.1.1.14.1.1 NResult.EArgument Field

Invalid argument.

C#

```
public const int EArgument = -10;
```

6.2.1.1.14.1.2 NResult.EArgumentNull Field

Argument is NULL (see page 133).

C#

```
public const int EArgumentNull = -11;
```

6.2.1.1.14.1.3 NResult.EArgumentOutOfRange Field

Argument is out of range.

C#

```
public const int EArgumentOutOfRange = -12;
```

6.2.1.1.14.1.4 NResult.EArithmetic Field

Arithmetic error occurred.

C#

```
public const int EArithmetic = -17;
```

6.2.1.1.14.1.5 NResult.EClr Field

CLR error occurred.

C#

```
public const int EClr = -93;
```

6.2.1.1.14.1.6 NResult.ECom Field

COM error occurred.

C#

```
public const int ECom = -92;
```

6.2.1.1.14.1.7 NResult.ECore Field

Core error occurred.

C#

```
public const int ECore = -2;
```

6.2.1.1.14.1.8 NResult.EDirectoryNotFound Field

Directory not found.

C#

```
public const int EDirectoryNotFound = -19;
```

6.2.1.1.14.1.9 NResult.EDriveNotFound Field

Drive not found.

C#

```
public const int EDriveNotFound = -20;
```

6.2.1.1.14.1.10 NResult.EEndOfStream Field

Unexpected end of stream.

C#

```
public const int EEndOfStream = -15;
```

6.2.1.1.14.1.11 NResult.EExternal Field

External error occurred.

C#

```
public const int EExternal = -90;
```

6.2.1.1.14.1.12 NResult.EFailed Field

Operation failed.

C#

```
public const int EFailed = -1;
```


6.2.1.1.14.1.13 NResult.EFileLoad Field

Error loading module from file.

C#

```
public const int EFileLoad = -22;
```

6.2.1.1.14.1.14 NResult.EFileNotFound Field

File not found.

C#

```
public const int EFileNotFound = -21;
```

6.2.1.1.14.1.15 NResult.EFormat Field

Argument is of invalid format.

C#

```
public const int EFormat = -13;
```

6.2.1.1.14.1.16 NResult.EIndexOutOfRange Field

Access with index was out of range.

C#

```
public const int EIndexOutOfRange = -9;
```

6.2.1.1.14.1.17 NResult.EInvalidCast Field

The cast is invalid.

C#

```
public const int EInvalidCast = -18;
```

6.2.1.1.14.1.18 NResult.EInvalidEnumArgument Field

Argument is an invalid enum value.

C#

```
public const int EInvalidEnumArgument = -16;
```

6.2.1.1.14.1.19 NResult.EInvalidOperation Field

Attempted to execute operation that is not valid for the object.

C#

```
public const int EInvalidOperation = -7;
```

6.2.1.1.14.1.20 NResult.EIO Field

IO (see page 885) error occurred.

C#

```
public const int EIO = -14;
```

6.2.1.1.14.1.21 NResult.ENotActivated Field

Operation is not activated.

C#

```
public const int ENotActivated = -200;
```

6.2.1.1.14.1.22 NResult.ENotImplemented Field

Operation is not implemented.

C#

```
public const int ENotImplemented = -5;
```

6.2.1.1.14.1.23 NResult.ENotSupported Field

Operation is not supported.

C#

```
public const int ENotSupported = -6;
```

6.2.1.1.14.1.24 NResult.ENullReference Field

Attempted to reference a NULL (see page 133) object.

C#

```
public const int ENullReference = -3;
```

6.2.1.1.14.1.25 NResult.EOutOfMemory Field

Out of memory.

C#

```
public const int EOutOfMemory = -4;
```

6.2.1.1.14.1.26 NResult.EOverflow Field

Overflow occurred.

C#

```
public const int EOverflow = -8;
```

6.2.1.1.14.1.27 NResult.EParameter Field

Invalid parameter Id.

C#

```
public const int EParameter = -100;
```

6.2.1.1.14.1.28 NResult.EParameterReadOnly Field

Attempted to set read-only parameter.

C#

```
public const int EParameterReadOnly = -101;
```

6.2.1.1.14.1.29 NResult.EPathTooLong Field

Path is too long.

C#

```
public const int EPathTooLong = -23;
```

6.2.1.1.14.1.30 NResult.ESecurity Field

Security error occurred.

C#

```
public const int ESecurity = -24;
```

6.2.1.1.14.1.31 NResult.ESys Field

Sys error occurred.

C#

```
public const int ESys = -94;
```

6.2.1.1.14.1.32 NResult.EWin32 Field

Win32 error occurred.

C#

```
public const int EWin32 = -91;
```

6.2.1.1.14.1.33 NResult.Ok Field

No error.

C#

```
public const int Ok = 0;
```

6.2.1.1.14.2 NResult Methods

6.2.1.1.14.2.1 NResult.Check Method

Checks if the result of a function indicates that an error has occurred and if so throws an exception.

C#

```
public static int Check(int result);
```

Parameters

Parameters	Description
int result	System.Int32 representing function's result.

6.2.1.1.14.2.2 NResult.IsFailed Method

Checks if the result of a function indicates that an error has occurred.

C#

```
public static bool IsFailed(int result);
```

Parameters

Parameters	Description
int result	System.Int32 representing function's result

Returns

True if an error has occurred, false otherwise.

6.2.1.1.14.2.3 NResult.IsSucceeded Method

Checks if the result of a function indicates that no error has occurred.

C#

```
public static bool IsSucceeded(int result);
```

Parameters

Parameters	Description
int result	System.Int32 representing function's result

Returns

True if no error has occurred, false otherwise.

6.2.1.1.14.2.4 NResult.RaiseError Method

Creates and throws exception object which represents the error.

C#

```
public static void RaiseError(int error);
```

Parameters

Parameters	Description
int error	Code of an error to be thrown.

6.2.1.1.14.2.5 NResult.SetError Method

For internal use only.

C#

```
public static int SetError(Exception error);
```

6.2.1.1.15 NURational Structure

Represents an unsigned rational number.

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct NURational {
    public static readonly NURational Empty = new NURational(0, 0);
}
```

NURational Fields

	Name	Description
	Empty (see page 636)	Represents a NURational that is a null reference.

Methods

	Name	Description
	NURational (see page 635)	Initializes a new instance of the NURational structure.

NURational Methods

	Name	Description
	ToString (see page 636)	Retrieves a string representation of the object.

NURational Properties

	Name	Description
	Denominator (see page 636)	Sets or retrieves the NRational (see page 628) value Denominator.
	Numerator (see page 636)	Sets or retrieves the NRational (see page 628) value Numerator.

6.2.1.1.15.1 NURational.NURational Constructor

Initializes a new instance of the NURational structure.

C#

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

Parameters

Parameters	Description
uint numerator	Numerator (see page 636) of this NURational.
uint denominator	Denominator (see page 636) of this NURational.

6.2.1.1.15.2 NURational Fields**6.2.1.1.15.2.1 NURational.Empty Field**

Represents a NURational ([see page 635](#)) that is a null reference.

C#

```
public static readonly NURational Empty = new NURational(0, 0);
```

6.2.1.1.15.3 NURational Methods**6.2.1.1.15.3.1 NURational.ToString Method**

Retrieves a string representation of the object.

C#

```
public override string ToString();
```

Returns

A string representation of an object.

6.2.1.1.15.4 NURational Properties**6.2.1.1.15.4.1 NURational.Denominator Property**

Sets or retrieves the NRational ([see page 628](#)) value Denominator.

C#

```
public uint Denominator;
```

6.2.1.1.15.4.2 NURational.Numerator Property

Sets or retrieves the NRational ([see page 628](#)) value Numerator.

C#

```
public uint Numerator;
```

6.2.1.1.16 ParameterException Class

This type of exception appears when trying to perform unsupported operation with the parameter.

Class Hierarchy**C#**

```
public class ParameterException : NeurotecExceptionBase;
```

INeurotecException Properties

	Name	Description
	Code (see page 638)	Gets an error code of the current exception.
	ManagedStackTrace (see page 638)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.

	UnmanagedStackTrace (see page 638)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

NeurotecExceptionBase Class

	Name	Description
	Code (see page 617)	Gets an error code of the current exception.
	ManagedStackTrace (see page 617)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	StackTrace (see page 617)	Gets a string representation of the frames on the call stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 617)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

6.2.1.1.17 ParameterReadOnlyException Class

This type of exception appears when trying to change parameter's value that is read only.

Class Hierarchy**C#**

```
public class ParameterReadOnlyException : ParameterException;
```

INeurotecException Properties

	Name	Description
	Code (see page 638)	Gets an error code of the current exception.
	ManagedStackTrace (see page 638)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 638)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

NeurotecExceptionBase Class

	Name	Description
	Code (see page 617)	Gets an error code of the current exception.
	ManagedStackTrace (see page 617)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	StackTrace (see page 617)	Gets a string representation of the frames on the call stack at the time the current exception was thrown.
	UnmanagedStackTrace (see page 617)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

6.2.1.2 Interfaces**6.2.1.2.1 INeurotecException Interface**

The interface that provides information about Neurotec ([see page 607](#)) exceptions.

Class Hierarchy**C#**

```
public interface INeurotecException;
```

INeurotecException Properties

	Name	Description
	Code (🔗 see page 638)	Gets an error code of the current exception.
	ManagedStackTrace (🔗 see page 638)	Gets a string representation of the frames on the managed stack at the time the current exception was thrown.
	UnmanagedStackTrace (🔗 see page 638)	Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

6.2.1.2.1.1 INeurotecException Properties

6.2.1.2.1.1.1 INeurotecException.Code Property

Gets an error code of the current exception.

C#

```
int Code;
```

6.2.1.2.1.1.2 INeurotecException.ManagedStackTrace Property

Gets a string representation of the frames on the managed stack at the time the current exception was thrown.

C#

```
string ManagedStackTrace;
```

6.2.1.2.1.1.3 INeurotecException.UnmanagedStackTrace Property

Gets a string representation of the frames on the unmanaged stack at the time the current exception was thrown.

C#

```
string UnmanagedStackTrace;
```

6.2.1.3 Structs, Records, Enums

6.2.1.3.1 Neurotec.NProcessorVendor Enumeration

Specifies the processor's vendor name.

C#

```
[Serializable]
public enum NProcessorVendor {
    Unknown = 0,
    Amd = 1,
    Centaur = 2,
    Cyrix = 3,
    Intel = 4,
    NationalSemiconductor = 5,
    NexGen = 6,
    RiseTechnology = 7,
    SiS = 8,
    Transmeta = 9,
    Umc = 10,
    Via = 11
}
```

Members

Members	Description
Unknown = 0	The processor's vendor is unknown.

Amd = 1	Advanced Micro Devices, Inc. (AMD).
Centaur = 2	Centaur Technology.
Cyrix = 3	Cyrix.
Intel = 4	Intel Corporation.
NationalSemiconductor = 5	National Semiconductor.
NexGen = 6	NexGen.
RiseTechnology = 7	Rise Technology.
SiS = 8	Silicon Integrated Systems (SIS) Corp.
Transmeta = 9	Transmeta Corporation.
Umc = 10	UMC.
Via = 11	VIA.

6.2.1.4 Types

6.2.1.4.1 Neurotec.NLibraryGetInfo Type

This delegate supports internal Neurotechnology libraries infrastructure and should not be used directly in your code.

C#

```
public delegate int NLibraryGetInfo(IntPtr pValue);
```

6.2.2 Neurotec.Biometrics Namespace

Classes under this namespace provides methods and properties for working with biometrical data.

Classes

	Name	Description
	NExtractors (see page 640)	Provides methods for retrieving NExtractors Library information.
	NFCore (see page 641)	The structure contains information of core.
	NFDelta (see page 643)	The structure contains information of delta.
	NFDoubleCore (see page 647)	The structure contains information of double core.
	NFExtractor (see page 648)	Provides functionality for extracting Neurotechnology Finger Records (NFRecords) from fingerprint images using algorithm encapsulated in Neurotechnology Fingerprint Features Extractor (NFExtractor) object.
	NFMinutia (see page 660)	The structure contains information of minutia.
	NFMinutiaNeighbor (see page 664)	The structure contains information of minutia neighbor.
	NFRecord (see page 666)	Provides functionality for packing, unpacking and editing Neurotechnology Finger Records (NFRecords).
	NFRecordInfo (see page 694)	For internal use.
	NFTemplate (see page 695)	Provides functionality for packing, unpacking and editing Neurotechnology Fingers Templates (NFTemplates).
	NFTemplateInfo (see page 712)	
	NleDetectionDetails (see page 714)	Structure containing face detection results.
	NleEyes (see page 714)	Structure defining information of an eye pair.
	NleFace (see page 715)	Structure containing information of a face location in an image.

	NleRotation (see page 716)	Structure containing information of a face rotation in an image as a deviation from frontal face in degrees.
	NLExtractor (see page 717)	Provides functionality for extracting Neurotechnology Face Records (NLRecord) from face images.
	NTemplate (see page 726)	Provides functionality for packing, unpacking and editing Neurotechnology Templates (NTemplates (see page 605)).
	NTemplateInfo (see page 744)	For internal use.
	NTemplates (see page 745)	Provides information about the NTemplates (see page 605) library.

Module

.NET Reference ([see page 606](#))

Structs, Records, Enums

	Name	Description
	NBiometricType (see page 746)	Specifies the type of biometric data.
	NfeExtractionStatus (see page 746)	Specifies fingerprint features extraction status.
	NfeReturnedImage (see page 747)	Specifies the returned image.
	NfeTemplateSize (see page 747)	Specifies a returned NFRecord (see page 666) size from methods: Extract, ExtractUnpacked.
	NFImpressionType (see page 747)	Specifies the impression types.
	NFMinutiaFormat (see page 748)	Specifies formats of minutia. This enumeration allows a bitwise combination of its member values.
	NFMinutiaOrder (see page 749)	Specifies minutia order.
	NFMinutiaType (see page 749)	Specifies types of minutia.
	NFPatternClass (see page 749)	Specifies pattern class of the fingerprint.
	NFPosition (see page 750)	Specifies finger position.
	NFRidgeCountsType (see page 752)	Specifies type of ridge counts contained in NFRecord (see page 666).
	NleExtractionStatus (see page 752)	Specifies face features extraction status.
	NleTemplateSize (see page 753)	Specifies NLRecord type by its size.

6.2.2.1 Classes

6.2.2.1.1 NExtractors Class

Provides methods for retrieving NExtractors Library information.

Class Hierarchy

C#

```
public static class NExtractors;
```

NExtractors Fields

	Name	Description
	DllName (see page 641)	Name of DLL containing unmanaged part of this class.

NExtractors Methods

	Name	Description
	GetInfo (see page 641)	Retrieves information about the library.

6.2.2.1.1.1 NExtractors Fields**6.2.2.1.1.1.1 NExtractors.DllName Field**

Name of DLL containing unmanaged part of this class.

C#

```
public const string DllName = "NExtractors";
```

6.2.2.1.1.2 NExtractors Methods**6.2.2.1.1.2.1 NExtractors.GetInfo Method**

Retrieves information about the library.

C#

```
public static NLibraryInfo GetInfo();
```

Returns

Returns NLibraryInfo ([see page 619](#)) structure with library information.

6.2.2.1.2 NFCore Structure

The structure contains information of core.

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct NFCore {
}
```

Methods

	Name	Description
	NFCore (see page 641)	Initializes a new instance of the NFCore structure.

NFCore Methods

	Name	Description
	ToString (see page 642)	Retrieves string representation of the specified object.

NFCore Properties

	Name	Description
	Angle (see page 642)	Gets or sets Angle of core.
	Y (see page 643)	Gets or sets x coordinate of core.
	RawAngle (see page 643)	Gets or sets raw angle of core
	X (see page 643)	Gets or sets y coordinate of core.

6.2.2.1.2.1 NFCore Constructor**6.2.2.1.2.1.1 NFCore.NFCore Constructor (ushort, ushort)**

Initializes a new instance of the NFCore structure.

C#

```
public NFCore(ushort x, ushort y);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the core.
ushort y	The y - coordinate of the core.

6.2.2.1.2.1.2 NFCore.NFCore Constructor (ushort, ushort, double)

Initializes a new instance of the NFCore class.

C#

```
public NFCore(ushort x, ushort y, double angle);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the core.
ushort y	The y - coordinate of the core.
double angle	The angle of the core.

6.2.2.1.2.1.3 NFCore.NFCore Constructor (ushort, ushort, int)

Initializes a new instance of the NFCore class.

C#

```
public NFCore(ushort x, ushort y, int angle);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the core.
ushort y	The y - coordinate of the core.
int angle	The angle of the core.

6.2.2.1.2.2 NFCore Methods**6.2.2.1.2.2.1 NFCore.ToString Method**

Retrieves string representation of the specified object.

C#

```
public override string ToString();
```

Returns

String representation of an object.

6.2.2.1.2.3 NFCore Properties**6.2.2.1.2.3.1 NFCore.Angle Property**

Gets or sets Angle of core.

C#

```
public double Angle;
```

Property value

The angle of the core.

6.2.2.1.2.3.2 NFCore.Y Property

Gets or sets x coordinate of core.

C#

```
public ushort Y;
```

Remarks

The y coordinate of the core is specified in pixels at resolution and "Y * [NFRecord (see page 666) vertical resolution] / Resolution" cannot be greater than MaxDimension or NFRecord (see page 666) width minus one.

Property value

The y coordinate of the core.

6.2.2.1.2.3.3 NFCore.RawAngle Property

Gets or sets raw angle of core

C#

```
public int RawAngle;
```

Remarks

The angle of the core is specified in 180/128 degrees units in counterclockwise order and cannot be less than zero or greater than 256 minus one. The value of -1 can be specified if the angle of the core is unknown.

Property value

The raw angle of the core.

6.2.2.1.2.3.4 NFCore.X Property

Gets or sets y coordinate of core.

C#

```
public ushort X;
```

Remarks

The x coordinate of the core is specified in pixels at resolution and "X * [NFRecord (see page 666) horizontal resolution] / Resolution" cannot be greater than MaxDimension or NFRecord (see page 666) width minus one.

Property value

The y coordinate of the core.

6.2.2.1.3 NFDelta Structure

The structure contains information of delta.

C#

```
[StructLayout(LayoutKind.Sequential)]  
public struct NFDelta {  
}
```

Methods

	Name	Description
	NFDelta (see page 644)	Initializes a new instance of the NFDelta structure.

NFDelta Methods

	Name	Description
	ToString (see page 645)	Returns a string that represents the current NFDelta object.

NFDelta Properties

	Name	Description
	Angle1 (see page 645)	Gets or sets the first angle of delta.
	Angle2 (see page 645)	Gets or sets the second angle of delta.
	Angle3 (see page 646)	Gets or sets the third angle of delta.
	Y (see page 646)	Gets or sets y coordinate of delta.
	RawAngle1 (see page 646)	Gets or sets the first row angle of the delta.
	RawAngle2 (see page 646)	Gets or sets the second row angle of the delta.
	RawAngle3 (see page 646)	Gets or sets the third row angle of the delta.
	X (see page 647)	Gets or sets x coordinate of delta.

6.2.2.1.3.1 NFDelta Constructor**6.2.2.1.3.1.1 NFDelta.NFDelta Constructor (ushort, ushort)**

Initializes a new instance of the NFDelta structure.

C#

```
public NFDelta(ushort x, ushort y);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the delta.
ushort y	The y - coordinate of the delta.

6.2.2.1.3.1.2 NFDelta.NFDelta Constructor (ushort, ushort, double, double, double)

Initializes a new instance of the NFDelta structure.

C#

```
public NFDelta(ushort x, ushort y, double angle1, double angle2, double angle3);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the delta.
ushort y	The y - coordinate of the delta.
double angle1	The first angle of the delta.
double angle2	The second angle of the delta.
double angle3	The third angle of the delta.

6.2.2.1.3.1.3 NFDelta.NFDelta Constructor (ushort, ushort, int, int, int)

Initializes a new instance of the NFDelta structure.

C#

```
public NFDelta(ushort x, ushort y, int angle1, int angle2, int angle3);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the delta.
ushort y	The y - coordinate of the delta.
int angle1	The first angle of the delta.
int angle2	The second angle of the delta.
int angle3	The third angle of the delta.

6.2.2.1.3.2 NFDelta Methods

6.2.2.1.3.2.1 NFDelta.ToString Method

Returns a string that represents the current NFDelta (see page 643) object.

C#

```
public override string ToString();
```

Returns

String that represents NFDelta (see page 643) object.

6.2.2.1.3.3 NFDelta Properties

6.2.2.1.3.3.1 NFDelta.Angle1 Property

Gets or sets the first angle of delta.

C#

```
public double Angle1;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and cannot be less than zero or greater than 256 minus one. The value of -1 can be specified if the first angle of the delta is unknown.

Property value

The first angle of the delta.

6.2.2.1.3.3.2 NFDelta.Angle2 Property

Gets or sets the second angle of delta.

C#

```
public double Angle2;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and cannot be less than zero or greater than 256 minus one. The value of -1 can be specified if the first angle of the delta is unknown.

Property value

The second angle of the delta.

6.2.2.1.3.3.3 NFDelta.Angle3 Property

Gets or sets the third angle of delta.

C#

```
public double Angle3;
```

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and cannot be less than zero or greater than 256 minus one. The value of -1 can be specified if the first angle of the delta is unknown.

Property value

The third angle of the delta.

6.2.2.1.3.3.4 NFDelta.Y Property

Gets or sets y coordinate of delta.

C#

```
public ushort Y;
```

Remarks

The y coordinate of the delta is specified in pixels at resolution and "Y * [NFRecord (see page 666) vertical resolution] / Resolution" cannot be greater than MaxDimension or NFRecord (see page 666) width minus one.

Property value

The x coordinate of the delta.

6.2.2.1.3.3.5 NFDelta.RawAngle1 Property

Gets or sets the first row angle of the delta.

C#

```
public int RawAngle1;
```

Property value

The first raw angle of the delta.

6.2.2.1.3.3.6 NFDelta.RawAngle2 Property

Gets or sets the second row angle of the delta.

C#

```
public int RawAngle2;
```

Property value

The second raw angle of the delta.

6.2.2.1.3.3.7 NFDelta.RawAngle3 Property

Gets or sets the third row angle of the delta.

C#

```
public int RawAngle3;
```

Property value

The third raw angle fog the delta.

6.2.2.1.3.3.8 NFDelta.X Property

Gets or sets x coordinate of delta.

C#

```
public ushort X;
```

Remarks

The x coordinate of the delta is specified in pixels at resolution and "X * [NFRecord (see page 666) horizontal resolution] / Resolution" cannot be greater than MaxDimension or NFRecord (see page 666) width minus one.

Property value

The x coordinate of the delta.

6.2.2.1.4 NFDoubleCore Structure

The structure contains information of double core.

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct NFDoubleCore {
}
```

Methods

	Name	Description
	NFDoubleCore (see page 647)	Initializes a new instance of the NFDoubleCore class.

NFDoubleCore Methods

	Name	Description
	ToString (see page 647)	Returns a string that represents the current NFDoubleCore object.

NFDoubleCore Properties

	Name	Description
	Y (see page 648)	Gets or sets y coordinate of the double core.
	X (see page 648)	Gets or sets x coordinate of the double core.

6.2.2.1.4.1 NFDoubleCore.NFDoubleCore Constructor

Initializes a new instance of the NFDoubleCore class.

C#

```
public NFDoubleCore(ushort x, ushort y);
```

Parameters

Parameters	Description
ushort x	The x coordinate of the double core.
ushort y	The y coordinate of the double core.

6.2.2.1.4.2 NFDoubleCore Methods

6.2.2.1.4.2.1 NFDoubleCore.ToString Method

Returns a string that represents the current NFDoubleCore (see page 647) object.

C#

```
public override string ToString();
```

Returns

String that represents NFDoubleCore ([↗](#) see page 647) object.

6.2.2.1.4.3 NFDoubleCore Properties**6.2.2.1.4.3.1 NFDoubleCore.Y Property**

Gets or sets y coordinate of the double core.

C#

```
public ushort Y;
```

Property value

The Y coordinate of the double core.

6.2.2.1.4.3.2 NFDoubleCore.X Property

Gets or sets x coordinate of the double core.

C#

```
public ushort X;
```

Property value

The X coordinate of the double core.

6.2.2.1.5 NFExtractor Class

Provides functionality for extracting Neurotechnology Finger Records (NFRecords) from fingerprint images using algorithm encapsulated in Neurotechnology Fingerprint Features Extractor (NFExtractor) object.

Class Hierarchy**C#**

```
public sealed class NFExtractor : NObject;
```

NFExtractor Fields**NFExtractor Class**

	Name	Description
	ModeAtmelFingerChip (↗ see page 651)	Atmel FingerChip scanner.
	ModeAuthentecAes2501B (↗ see page 651)	AuthenTec AES2501B scanner.
	ModeAuthentecAes4000 (↗ see page 652)	AuthenTec AES4000 scanner.
	ModeAuthentecAFS2 (↗ see page 652)	AuthenTec AF-S2 scanner.
	ModeBiometrikaFX2000 (↗ see page 652)	BiometriKa FX 2000 scanner.
	ModeBiometrikaFX3000 (↗ see page 652)	BiometriKa FX 3000 scanner.
	ModeBmfBlp100 (↗ see page 652)	BMF BLP100 scanner.

ModeCrossMatchVerifier300 (see page 652)	Cross Match Verifier 300 scanner.
ModeDigentIzzix (see page 652)	Digent Izzix 1000 scanner.
ModeDigitalPersonaUareU (see page 652)	DigitalPersona U.are.U 2000 / 4000 / 4500 scanners.
ModeEthentica (see page 652)	Ethentica scanner.
ModeFujitsuMbf200 (see page 653)	Fujitsu MBF200 scanner.
ModeFutronicFS80 (see page 653)	Futronic FS80 scanner.
ModeGeneral (see page 653)	This mode is used when none of these ModeXXX modes suites.
ModeIdenticatorTechnologyDF90 (see page 653)	Identicator Technology DF90 scanner.
ModeIdentixDfr2090 (see page 653)	Identix DFR 2090 scanner.
ModeIdentixTouchView (see page 653)	Identix TouchView scanner.
ModeKeytronicSecureDesktop (see page 653)	Keytronic Secure Desktop scanner.
ModeLighTuningLttC500 (see page 653)	LighTuning LTT-C500 Fingerprint Sensor.
ModeNitgenFingkeyHamster (see page 653)	NITGEN Fingkey Hamster scanner.
ModePreciseBiometrics100CS (see page 654)	Precise Biometrics (see page 639) 100CS scanner.
ModeSecugenHamster (see page 654)	SecuGen Hamster scanner.
ModeStartekFM200 (see page 654)	Startek FM200 scanner.
ModeTacomaCmos (see page 654)	Tacoma CMOS scanner.
ModeTestechBioI (see page 654)	Testech Bio-i scanner.
ModeUpekTouchChip (see page 654)	UPEK TouchChip.
ParameterExtractedRidgeCounts (see page 654)	Identifier specifying type of ridge counts extracted parameter of type int. Parameter value can be one of the NFRidgeCountsType (see page 752) enumeration members.
ParameterGeneralizationMaximalRotation (see page 654)	Maximal rotation of two features collection to each other. Must be in range 0°..180°.
ParameterGeneralizationMinMinutiaCount (see page 654)	Identifier specifying minimal minutiae count for generalization.
ParameterGeneralizationThreshold (see page 655)	Has the same meaning for features generalization as ParameterMatchingThreshold parameter for features matching.
ParameterMinMinutiaCount (see page 655)	Identifier specifying minimal minutiae count.
ParameterMode (see page 655)	Identifier specifying mode (parameter value set) parameter of type uint. Parameter value can be one of the ModeXXX.
ParameterQualityThreshold (see page 655)	Identifier specifies image quality threshold.
ParameterReturnedImage (see page 655)	Identifier specifying kind of image returned after extraction parameter of type int. Parameter value can be one of the NfeReturnedImage (see page 747) enumeration members.

	ParameterTemplateSize (see page 655)	Identifier specifying template size parameter. Parameter value can be one of the TemplateSize (see page 660) enumeration members.
	ParameterUseQuality (see page 655)	If identifier is set to true, then methods Extract (see page 655), ExtractUnpacked determines the quality of the image. If the image quality is lower than ParameterQualityThreshold (see page 655) the method do not returns template.

Methods

NFExtractor Class

	Name	Description
	NFExtractor (see page 651)	Initializes a new instance of the NFExtractor class.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
 	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
 	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NFExtractor Class

	Name	Description
	Extract (see page 655)	Extracts a NFRecord (see page 666) from a fingerprint image.
	Generalize (see page 656)	Generalizes count features collections to single features collection.
 	GetStaticParameter (see page 656)	Retrieves static parameter by parameter identifier.
 	IsTemplateSmall (see page 657)	Checks if specified NFRecord (see page 666) is of size NfeTemplateSize.Small.
 	MakeLightTemplate (see page 657)	Makes template for FingerCell matcher.
 	MakeSmallTemplate (see page 657)	Makes template for FingerCell matcher of NfeTemplateSize.Small size.
 	SetStaticParameter (see page 658)	Sets the static parameter by parameter identifier.
	UpdateTemplate (see page 658)	

NObject Properties

NObject Class

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NFExtractor Class

	Name	Description
	ExtractedRidgeCounts (see page 658)	Gets or sets extracted ridge counts.
	GeneralizationMaximalRotation (see page 658)	Gets or sets maximal rotation of two features collection to each other.
	GeneralizationMinMinutiaCount (see page 659)	Gets or sets minimal minutiae count required for generalization.
	GeneralizationThreshold (see page 659)	Gets or sets generalization threshold.
	IsPalm (see page 659)	Checks if the NFExtractor is a palmprint extractor.
	MinMinutiaCount (see page 659)	Gets or sets minimal minutiae count.
	Mode (see page 659)	Gets or sets scanners mode.
	QualityThreshold (see page 660)	Gets or sets image quality threshold.
	ReturnedImage (see page 660)	Gets or sets the image type.
	TemplateSize (see page 660)	Gets or sets template size.
	UseQuality (see page 660)	Gets or sets value indicating whether image quality should be checked.

6.2.2.1.5.1 NFExtractor Constructor**6.2.2.1.5.1.1 NFExtractor.NFExtractor Constructor ()**

Initializes a new instance of the NFExtractor class.

C#

```
public NFExtractor();
```

6.2.2.1.5.1.2 NFExtractor.NFExtractor Constructor (bool)

Initializes a new instance of the NFExtractor class.

C#

```
public NFExtractor(bool isPalm);
```

Parameters

Parameters	Description
bool isPalm	NFExtractor is a palmprint extractor.

6.2.2.1.5.2 NFExtractor Fields**6.2.2.1.5.2.1 NFExtractor.ModeAtmelFingerChip Field**

Atmel FingerChip scanner.

C#

```
public const uint ModeAtmelFingerChip = 900;
```

6.2.2.1.5.2.2 NFExtractor.ModeAuthentecAes2501B Field

AuthenTec AES2501B scanner.

C#

```
public const uint ModeAuthentecAes2501B = 811;
```

6.2.2.1.5.2.3 NExtractor.ModeAuthentecAes4000 Field

AuthenTec AES4000 scanner.

C#

```
public const uint ModeAuthentecAes4000 = 810;
```

6.2.2.1.5.2.4 NExtractor.ModeAuthentecAFS2 Field

AuthenTec AF-S2 scanner.

C#

```
public const uint ModeAuthentecAFS2 = 800;
```

6.2.2.1.5.2.5 NExtractor.ModeBiometrikaFX2000 Field

BiometriKa FX 2000 scanner.

C#

```
public const uint ModeBiometrikaFX2000 = 200;
```

6.2.2.1.5.2.6 NExtractor.ModeBiometrikaFX3000 Field

BiometriKa FX 3000 scanner.

C#

```
public const uint ModeBiometrikaFX3000 = 201;
```

6.2.2.1.5.2.7 NExtractor.ModeBmfBlp100 Field

BMF BLP100 scanner.

C#

```
public const uint ModeBmfBlp100 = 1000;
```

6.2.2.1.5.2.8 NExtractor.ModeCrossMatchVerifier300 Field

Cross Match Verifier 300 scanner.

C#

```
public const uint ModeCrossMatchVerifier300 = 1300;
```

6.2.2.1.5.2.9 NExtractor.ModeDigentIzzix Field

Digent Izzix 1000 scanner.

C#

```
public const uint ModeDigentIzzix = 1600;
```

6.2.2.1.5.2.10 NExtractor.ModeDigitalPersonaUareU Field

DigitalPersona U.are.U 2000 / 4000 / 4500 scanners.

C#

```
public const uint ModeDigitalPersonaUareU = 100;
```

6.2.2.1.5.2.11 NExtractor.ModeEthentica Field

Ethentica scanner.

C#

```
public const uint ModeEthentica = 1200;
```

6.2.2.1.5.2.12 NExtractor.ModeFujitsuMbf200 Field

Fujitsu MBF200 scanner.

C#

```
public const uint ModeFujitsuMbf200 = 1800;
```

6.2.2.1.5.2.13 NExtractor.ModeFutronicFS80 Field

Futronic FS80 scanner.

C#

```
public const uint ModeFutronicFS80 = 1900;
```

6.2.2.1.5.2.14 NExtractor.ModeGeneral Field

This mode is used when none of these ModeXXX modes suites.

C#

```
public const uint ModeGeneral = 0;
```

6.2.2.1.5.2.15 NExtractor.ModeIdenticatorTechnologyDF90 Field

Identicator Technology DF90 scanner.

C#

```
public const uint ModeIdenticatorTechnologyDF90 = 700;
```

6.2.2.1.5.2.16 NExtractor.ModeIdentixDfr2090 Field

Identix DFR 2090 scanner.

C#

```
public const uint ModeIdentixDfr2090 = 401;
```

6.2.2.1.5.2.17 NExtractor.ModeIdentixTouchView Field

Identix TouchView scanner.

C#

```
public const uint ModeIdentixTouchView = 400;
```

6.2.2.1.5.2.18 NExtractor.ModeKeytronicSecureDesktop Field

Keytronic Secure Desktop scanner.

C#

```
public const uint ModeKeytronicSecureDesktop = 300;
```

6.2.2.1.5.2.19 NExtractor.ModeLighTuningLttC500 Field

LighTuning LTT-C500 Fingerprint Sensor.

C#

```
public const uint ModeLighTuningLttC500 = 2000;
```

6.2.2.1.5.2.20 NExtractor.ModeNitgenFingkeyHamster Field

NITGEN Fingkey Hamster scanner.

C#

```
public const uint ModeNitgenFingkeyHamster = 1400;
```

6.2.2.1.5.2.21 NExtractor.ModePreciseBiometrics100CS Field

Precise Biometrics (see page 639) 100CS scanner.

C#

```
public const uint ModePreciseBiometrics100CS = 500;
```

6.2.2.1.5.2.22 NExtractor.ModeSecugenHamster Field

SecuGen Hamster scanner.

C#

```
public const uint ModeSecugenHamster = 1100;
```

6.2.2.1.5.2.23 NExtractor.ModeStartekFM200 Field

Startek FM200 scanner.

C#

```
public const uint ModeStartekFM200 = 1700;
```

6.2.2.1.5.2.24 NExtractor.ModeTacomaCmos Field

Tacoma CMOS scanner.

C#

```
public const uint ModeTacomaCmos = 2100;
```

6.2.2.1.5.2.25 NExtractor.ModeTestechBioI Field

Testech Bio-i scanner.

C#

```
public const uint ModeTestechBioI = 1500;
```

6.2.2.1.5.2.26 NExtractor.ModeUpekTouchChip Field

UPEK TouchChip.

C#

```
public const uint ModeUpekTouchChip = 600;
```

6.2.2.1.5.2.27 NExtractor.ParameterExtractedRidgeCounts Field

Identifier specifying type of ridge counts extracted parameter of type int. Parameter value can be one of the NFRidgeCountsType (see page 752) enumeration members.

C#

```
public const ushort ParameterExtractedRidgeCounts = 120;
```

6.2.2.1.5.2.28 NExtractor.ParameterGeneralizationMaximalRotation Field

Maximal rotation of two features collection to each other. Must be in range 0°..180°.

C#

```
public const ushort ParameterGeneralizationMaximalRotation = 301;
```

6.2.2.1.5.2.29 NExtractor.ParameterGeneralizationMinMinutiaCount Field

Identifier specifying minimal minutiae count for generalization.

C#

```
public const ushort ParameterGeneralizationMinMinutiaCount = 319;
```

6.2.2.1.5.2.30 NExtractor.ParameterGeneralizationThreshold Field

Has the same meaning for features generalization as ParameterMatchingThreshold parameter for features matching.

C#

```
public const ushort ParameterGeneralizationThreshold = 300;
```

6.2.2.1.5.2.31 NExtractor.ParameterMinMinutiaCount Field

Identifier specifying minimal minutiae count.

C#

```
public const ushort ParameterMinMinutiaCount = 119;
```

6.2.2.1.5.2.32 NExtractor.ParameterMode Field

Identifier specifying mode (parameter value set) parameter of type uint. Parameter value can be one of the ModeXXX.

C#

```
public const ushort ParameterMode = 1000;
```

6.2.2.1.5.2.33 NExtractor.ParameterQualityThreshold Field

Identifier specifies image quality threshold.

C#

```
public const ushort ParameterQualityThreshold = 910;
```

6.2.2.1.5.2.34 NExtractor.ParameterReturnedImage Field

Identifier specifying kind of image returned after extraction parameter of type int. Parameter value can be one of the NfeReturnedImage (see page 747) enumeration members.

C#

```
public const ushort ParameterReturnedImage = 190;
```

6.2.2.1.5.2.35 NExtractor.ParameterTemplateSize Field

Identifier specifying template size parameter. Parameter value can be one of the TemplateSize (see page 660) enumeration members.

C#

```
public const ushort ParameterTemplateSize = 100;
```

6.2.2.1.5.2.36 NExtractor.ParameterUseQuality Field

If identifier is set to true, then methods Extract (see page 655), ExtractUnpacked determines the quality of the image. If the image quality is lower than ParameterQualityThreshold (see page 655) the method do not returns template.

C#

```
public const ushort ParameterUseQuality = 900;
```

6.2.2.1.5.3 NExtractor Methods**6.2.2.1.5.3.1 NExtractor.Extract Method**

Extracts a NRecord (see page 666) from a fingerprint image.

C#

```
public NRecord Extract(NGrayscaleImage image, NPosition position, NImpressionType impressionType, out NfeExtractionStatus extractionStatus);
```


Parameters

Parameters	Description
NGrayscaleImage image	The NGrayscaleImage (see page 832) image.
NFPosition position	The finger NFPosition (see page 750).
NFImpressionType impressionType	The finger NFImpressionType (see page 747).
out NfeExtractionStatus extractionStatus	One of the NfeExtractionStatus (see page 746) values.

Returns

NFRecord ([see page 666](#)) object.

Remarks

position and impressionType are written to extracted NFRecord ([see page 666](#)). Method accepts images with 50-5000 ppi. If image resolution is not equal 500 ppi it will be converted to 500 ppi. Recommended image resolution is 500 ppi.

6.2.2.1.5.3.2 Generalize Method**6.2.2.1.5.3.2.1 NFExtractor.Generalize Method (NFRecord[], out NfeExtractionStatus, out int)**

Generalizes count features collections to single features collection.

C#

```
public NFRecord Generalize(NFRecord[] records, out NfeExtractionStatus status, out int baseTemplateIndex);
```

Parameters

Parameters	Description
NFRecord[] records	The array of NFRecord (see page 666) templates.
out NfeExtractionStatus status	Generalization status
out int baseTemplateIndex	Index of main generalization template.

Returns

Byte array that contains generalized NFRecord ([see page 666](#)).

6.2.2.1.5.3.2.2 NFExtractor.Generalize Method (NFRecord[], out int)

Generalizes count features collections to single features collection.

C#

```
[Obsolete("Use Generalize(NFRecord[], NfeExtractionStatus, int) instead.")]  
public NFRecord Generalize(NFRecord[] records, out int baseTemplateIndex);
```

Parameters

Parameters	Description
NFRecord[] records	The array of NFRecord (see page 666) templates.
out int baseTemplateIndex	Index of main generalization template.

Returns

Byte array that contains generalized NFRecord ([see page 666](#)).

Remarks

Method is obsolete. Method is obsolete, use the Generalize(NFRecord ([see page 666](#))[], out int) instead.

6.2.2.1.5.3.3 NFExtractor.GetStaticParameter Method

Retrieves static parameter by parameter identifier.

C#

```
[Obsolete("Use NativeType.GetParameter instead.")]  
public static object GetStaticParameter(ushort parameterId);
```

Parameters

Parameters	Description
ushort parameterId	The static parameter identifier.

Returns

The parameter value.

Remarks

Unused method.

6.2.2.1.5.3.4 NFExtractor.IsTemplateSmall Method

Checks if specified NFRecord (see page 666) is of size NfeTemplateSize.Small.

C#

```
public static bool IsTemplateSmall(NFRecord record);
```

Parameters

Parameters	Description
NFRecord record	NFRecord (see page 666) object.

Returns

True if NFRecord (see page 666) is of size NfeTemplateSize.Small, false otherwise.

6.2.2.1.5.3.5 NFExtractor.MakeLightTemplate Method

Makes template for FingerCell matcher.

C#

```
public static NFRecord MakeLightTemplate(NFRecord record, bool makeSmallTemplate, bool  
preserveRidgeCounts);
```

Parameters

Parameters	Description
NFRecord record	NFRecord (see page 666) object.
bool makeSmallTemplate	Determinates template size. If parameter will be true then method returns NfeTemplateSize.Small NFRecord (see page 666) FingerCell template; otherwise, NfeTemplateSize.Large NFRecord (see page 666) FingerCell template.
bool preserveRidgeCounts	Reserved. Should be false.

Returns

NFRecord (see page 666) template for FingerCell matcher.

Remarks

makeSmallTemplate should be NfeTemplateSize.Small for FingerCell matcher.

6.2.2.1.5.3.6 NFExtractor.MakeSmallTemplate Method

Makes template for FingerCell matcher of NfeTemplateSize.Small size.

C#

```
public static NFRecord MakeSmallTemplate(NFRecord record);
```

Parameters

Parameters	Description
NFRecord record	NFRecord (see page 666) object.

Returns

NFRecord ([see page 666](#)) template for FingerCell matcher.

6.2.2.1.5.3.7 NFExtractor.SetStaticParameter Method

Sets the static parameter by parameter identifier.

C#

```
[Obsolete("Use NativeType.SetParameter instead.")]  
public static void SetStaticParameter(ushort parameterId, object value);
```

Parameters

Parameters	Description
ushort parameterId	The static parameter identifier.
object value	The static parameter value.

Remarks

Unused method.

6.2.2.1.5.3.8 NFExtractor.UpdateTemplate Method**C#**

```
public void UpdateTemplate(NGrayscaleImage image, NFRecord record);
```

Parameters

Parameters	Description
NGrayscaleImage image	The NGrayscaleImage (see page 832) image.
NFRecord record	The NFRecord (see page 666) object.

Remarks

This method updates ridge counts and minutiae curvatures and Gs using the image provided.

6.2.2.1.5.4 NFExtractor Properties**6.2.2.1.5.4.1 NFExtractor.ExtractedRidgeCounts Property**

Gets or sets extracted ridge counts.

C#

```
public NFRidgeCountsType ExtractedRidgeCounts;
```

Property value

One of the NFRidgeCountsType ([see page 752](#)) values.

6.2.2.1.5.4.2 NFExtractor.GeneralizationMaximalRotation Property

Gets or sets maximal rotation of two features collection to each other.

C#

```
public byte GeneralizationMaximalRotation;
```

Remarks

Maximal rotation of two features collection to each other. Must be in range 0°..180°. This parameter is important to Generalize (see page 656), GeneralizeUnpacked methods. The generalization maximal rotation can be set also through SetParameter (see page 623) method.

Property value

Gets or sets generalization maximal rotation.

6.2.2.1.5.4.3 NFExtractor.GeneralizationMinMinutiaCount Property

Gets or sets minimal minutiae count required for generalization.

C#

```
public int GeneralizationMinMinutiaCount;
```

Property value

Gets or sets generalization minimal minutiae count.

6.2.2.1.5.4.4 NFExtractor.GeneralizationThreshold Property

Gets or sets generalization threshold.

C#

```
public int GeneralizationThreshold;
```

Remarks

Has the same meaning for features generalization as NFMP_MATCHING_THRESHOLD parameter for features matching. The generalization threshold can be set also through SetParameter (see page 623) method.

Property value

Get or set generalization threshold.

6.2.2.1.5.4.5 NFExtractor.IsPalm Property

Checks if the NFExtractor (see page 648) is a palmprint extractor.

C#

```
public bool IsPalm;
```

Property value

true if NFExtractor (see page 648) is a palmprint extractor; otherwise, false.

6.2.2.1.5.4.6 NFExtractor.MinMinutiaCount Property

Gets or sets minimal minutiae count.

C#

```
public int MinMinutiaCount;
```

Property value

Gets or sets minimal minutiae count.

6.2.2.1.5.4.7 NFExtractor.Mode Property

Gets or sets scanners mode.

C#

```
public uint Mode;
```

Property value

One of the Mode constants.

6.2.2.1.5.4.8 NExtractor.QualityThreshold Property

Gets or sets image quality threshold.

C#

```
public byte QualityThreshold;
```

Remarks

The image quality threshold can be set also through SetParameter (see page 623) method.

Property value

The image quality threshold. The value should be in range [0, 255].

6.2.2.1.5.4.9 NExtractor.ReturnedImage Property

Gets or sets the image type.

C#

```
public NfeReturnedImage ReturnedImage;
```

Property value

One of the NfeReturnedImage (see page 747) value.

6.2.2.1.5.4.10 NExtractor.TemplateSize Property

Gets or sets template size.

C#

```
public NfeTemplateSize TemplateSize;
```

Property value

One of the NfeTemplateSize (see page 747) value.

6.2.2.1.5.4.11 NExtractor.UseQuality Property

Gets or sets value indicating whether image quality should be checked.

C#

```
public bool UseQuality;
```

Remarks

If property returns true, then methods Extract (see page 655), ExtractUnpacked determines the quality of the image. If the image quality is lower than ParameterQualityThreshold (see page 655) the method do not returns template. The using of the quality can be set also through SetParameter (see page 623) method.

Property value

true if then methods Extract (see page 655), ExtractUnpacked determines the quality of the image.

6.2.2.1.6 NFMinutia Structure

The structure contains information of minutia.

C#

```
[StructLayout(LayoutKind.Sequential)]  
public struct NFMinutia {  
}
```

Methods

	Name	Description
	NFMinutia (see page 661)	Initializes a new instance of the NFMinutia class.

NFMinutia Methods

	Name	Description
	ToString (see page 662)	Returns a string that represents the current NFMinutia object.

NFMinutia Properties

	Name	Description
	Angle (see page 662)	Gets or sets the angle of the minutia.
	Curvature (see page 663)	Gets or sets the ridge curvature near minutia.
	G (see page 663)	Gets or sets the G (ridge density) near minutia.
	Y (see page 663)	Gets or sets y coordinate of the minutia.
	Quality (see page 663)	Gets or sets quality of the minutia.
	RawAngle (see page 664)	Gets or sets the raw angle of the minutia.
	Type (see page 664)	Gets or sets the type of the minutia.
	X (see page 664)	Gets or sets x coordinate of the minutia.

6.2.2.1.6.1 NFMinutia Constructor**6.2.2.1.6.1.1 NFMinutia.NFMinutia Constructor (ushort, ushort, NFMinutiaType, byte)**

Initializes a new instance of the NFMinutia class.

C#

```
public NFMinutia(ushort x, ushort y, NFMinutiaType type, byte angle);
```

Parameters

Parameters	Description
ushort x	The x coordinate of the minutia.
ushort y	The y coordinate of the minutia.
NFMinutiaType type	One of the NFMinutiaType (see page 749) values.
byte angle	The angle of the minutia.

6.2.2.1.6.1.2 NFMinutia.NFMinutia Constructor (ushort, ushort, NFMinutiaType, byte, byte, byte, byte)

Initializes a new instance of the NFMinutia class.

C#

```
public NFMinutia(ushort x, ushort y, NFMinutiaType type, byte angle, byte quality, byte curvature, byte g);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the minutia.
ushort y	The y - coordinate of the minutia.
NFMinutiaType type	One of the NFMinutiaType (see page 749) values.

byte angle	The angle of the minutia.
byte quality	The quality of the minutia.
byte curvature	The ridge curvature near minutia.
byte g	The G (see page 663) (ridge density) near minutia.

6.2.2.1.6.1.3 NMinutia.NMinutia Constructor (ushort, ushort, NMinutiaType, double)

Initializes a new instance of the NMinutia class.

C#

```
public NMinutia(ushort x, ushort y, NMinutiaType type, double angle);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the minutia.
ushort y	The y - coordinate of the minutia.
NMinutiaType type	One of the NMinutiaType (see page 749) values.
double angle	The angle of the minutia.

6.2.2.1.6.1.4 NMinutia.NMinutia Constructor (ushort, ushort, NMinutiaType, double, byte, byte, byte)

Initializes a new instance of the NMinutia class.

C#

```
public NMinutia(ushort x, ushort y, NMinutiaType type, double angle, byte quality, byte curvature, byte g);
```

Parameters

Parameters	Description
ushort x	The x - coordinate of the minutia.
ushort y	The y - coordinate of the minutia.
NMinutiaType type	One of the NMinutiaType (see page 749) values.
double angle	The angle of the minutia.
byte quality	The quality of the minutia.
byte curvature	The ridge curvature near minutia.
byte g	The G (see page 663) (ridge density) near minutia.

6.2.2.1.6.2 NMinutia Methods

6.2.2.1.6.2.1 NMinutia.ToString Method

Returns a string that represents the current NMinutia (see page 660) object.

C#

```
public override string ToString();
```

Returns

String representation of NMinutia (see page 660) object.

6.2.2.1.6.3 NMinutia Properties

6.2.2.1.6.3.1 NMinutia.Angle Property

Gets or sets the angle of the minutia.

C#

```
public double Angle;
```

Remarks

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

Property value

The angle of the minutia.

6.2.2.1.6.3.2 NMinutia.Curvature Property

Gets or sets the ridge curvature near minutia.

C#

```
public byte Curvature;
```

Remarks

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

Property value

The ridge curvature near minutia.

6.2.2.1.6.3.3 NMinutia.G Property

Gets or sets the G (ridge density) near minutia.

C#

```
public byte G;
```

Remarks

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

Property value

The G (ridge density) near minutia.

6.2.2.1.6.3.4 NMinutia.Y Property

Gets or sets y coordinate of the minutia.

C#

```
public ushort Y;
```

Remarks

The y coordinate of the minutia is specified in pixels at resolution and "Y * [NFRecord (see page 666) vertical resolution] / Resolution" cannot be greater than MaxDimension or NFRecord (see page 666) width minus one.

Property value

The Y coordinate of the minutia.

6.2.2.1.6.3.5 NMinutia.Quality Property

Gets or sets quality of the minutia.

C#

```
public byte Quality;
```


Remarks

The quality of the minutia must be in the range [0, 100].The higher it is, the better the quality of the minutia is. If quality of the minutia is unknown it must be set to zero.

Property value

The quality of the minutia.

6.2.2.1.6.3.6 NMinutia.RawAngle Property

Gets or sets the raw angle of the minutia.

C#

```
public byte RawAngle;
```

Property value

The raw angle of the minutia.

6.2.2.1.6.3.7 NMinutia.Type Property

Gets or sets the type of the minutia.

C#

```
public NMinutiaType Type;
```

Property value

One of the NMinutiaType (see page 749) values.

6.2.2.1.6.3.8 NMinutia.X Property

Gets or sets x coordinate of the minutia.

C#

```
public ushort X;
```

Remarks

The x coordinate of the minutia is specified in pixels at resolution and "X * [NRecord horizontal resolution] / Resolution" cannot be greater than MaxDimension or NRecord width minus one.

Property value

The X coordinate of the minutia.

6.2.2.1.7 NMinutiaNeighbor Structure

The structure contains information of minutia neighbor.

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct NMinutiaNeighbor {
    public static readonly NMinutiaNeighbor Empty = new NMinutiaNeighbor(-1, 255);
}
```

NMinutiaNeighbor Fields

	Name	Description
	Empty (see page 665)	Represents a NMinutiaNeighbor that is a null reference.

Methods

	Name	Description
	NFMinutiaNeighbor (see page 665)	Initializes a new instance of the NFMinutiaNeighbor class.

NFMinutiaNeighbor Methods

	Name	Description
	ToString (see page 665)	Returns a string that represents the current NFMinutiaNeighbor object.

NFMinutiaNeighbor Properties

	Name	Description
	Index (see page 665)	Gets or sets the index of minutia neighbor.
	RidgeCount (see page 666)	Gets or sets the ridge count between the minutia and minutia neighbor.

6.2.2.1.7.1 NFMinutiaNeighbor.NFMinutiaNeighbor Constructor

Initializes a new instance of the NFMinutiaNeighbor class.

C#

```
public NFMinutiaNeighbor(int index, byte ridgeCount);
```

Parameters

Parameters	Description
int index	The index of neighbor.
byte ridgeCount	The ridge count of neighbor.

6.2.2.1.7.2 NFMinutiaNeighbor Fields**6.2.2.1.7.2.1 NFMinutiaNeighbor.Empty Field**

Represents a NFMinutiaNeighbor (see page 664) that is a null reference.

C#

```
public static readonly NFMinutiaNeighbor Empty = new NFMinutiaNeighbor(-1, 255);
```

6.2.2.1.7.3 NFMinutiaNeighbor Methods**6.2.2.1.7.3.1 NFMinutiaNeighbor.ToString Method**

Returns a string that represents the current NFMinutiaNeighbor (see page 664) object.

C#

```
public override string ToString();
```

Returns

String representation of NFMinutiaNeighbor (see page 664) object.

6.2.2.1.7.4 NFMinutiaNeighbor Properties**6.2.2.1.7.4.1 NFMinutiaNeighbor.Index Property**

Gets or sets the index of minutia neighbor.

C#

```
public int Index;
```

Property value

The index of minutia neighbor.

6.2.2.1.7.4.2 NfMinutiaNeighbor.RidgeCount Property

Gets or sets the ridge count between the minutia and minutia neighbor.

C#

```
public byte RidgeCount;
```

Property value

The ridge count between the minutia and minutia neighbor.

6.2.2.1.8 NFRecord Class

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

Class Hierarchy**C#**

```
public sealed class NFRecord : NObject, ICloneable;
```

NFRecord Classes**NFRecord Class**

	Name	Description
	CoreCollection (see page 672)	Represents the collection of NFCore (see page 641).
	DeltaCollection (see page 673)	Represents the collection of NFDelta (see page 643).
	DoubleCoreCollection (see page 673)	Represents the collection of NFDoubleCore (see page 647).
	MinutiaCollection (see page 673)	Represents the collection of NFMinutia (see page 660).
	MinutiaNeighborsCollection (see page 673)	Represents the collection of NFMinutiaNeighbor (see page 664).

NFRecord Fields**NFRecord Class**

	Name	Description
	FlagSaveBlockedOrients (see page 673)	The flag indicating whether blocked orientations should be packed in NFRecord.
	FlagSaveV2 (see page 673)	The flag indicating whether NFRecord should be saved using version 2 algorithm.
	FlagSaveV3 (see page 673)	The flag indicating whether NFRecord should be saved using version 3 algorithm.
	FlagSkipBlockedOrients (see page 674)	The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.
	FlagSkipCurvatures (see page 674)	The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.
	FlagSkipGs (see page 674)	The flag indicating whether G (see page 692) field should be skipped while unpacking or packing NFRecord.
	FlagSkipQualities (see page 674)	The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.
	FlagSkipRidgeCounts (see page 674)	The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.
	FlagSkipSingularPoints (see page 674)	The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.

	MaxCoreCount (see page 674)	The maximum number of cores a NFRecord can contain.
	MaxDeltaCount (see page 674)	The maximum number of deltas a NFRecord can contain.
	MaxDimension (see page 674)	The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.
	MaxDoubleCoreCount (see page 675)	The maximum number of double cores a NFRecord can contain.
	MaxFingerCoreCount (see page 675)	The maximum number of cores a NFRecord can contain.
	MaxFingerDeltaCount (see page 675)	The maximum number of deltas a NFRecord can contain.
	MaxFingerDimension (see page 675)	The maximum number of finger dimension.
	MaxFingerDoubleCoreCount (see page 675)	The maximum number of double cores a NFRecord can contain.
	MaxFingerMinutiaCount (see page 675)	The maximum number of finger minutiae a NFRecord can contain.
	MaxMinutiaCount (see page 675)	The maximum number of minutiae a NFRecord can contain.
	MaxPalmCoreCount (see page 675)	The maximum number of cores in a palmprint a NFRecord can contain.
	MaxPalmDeltaCount (see page 676)	The maximum number of deltas in a palmprint a NFRecord can contain.
	MaxPalmDimension (see page 676)	The maximum number of a palmprint dimension.
	MaxPalmDoubleCoreCount (see page 676)	The maximum number of double cores in a palmprint a NFRecord can contain.
	MaxPalmMinutiaCount (see page 676)	The maximum number of minutiae in a palmprint a NFRecord can contain.
	Resolution (see page 676)	The resolution of minutiae, cores, deltas and double cores coordinates in a NFRecord.

Methods

NFRecord Class

	Name	Description
	NFRecord (see page 669)	Initializes a new instance of the NFRecord class.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NFRecord Class

	Name	Description
	Check (see page 676)	Checks if format of packed NFRecord is correct.
	Clone (see page 677)	Creates NFRecord object from another NFRecord object.
	FromHandle (see page 677)	Creates NFRecord object from handle.
	GetCbeffProductType (see page 677)	Retrieves the Cbeff product type of the NFRecord.
	GetG (see page 678)	Retrieves G (see page 692) from packed NFRecord.
	GetHeight (see page 679)	Retrieves height of fingerprint image from packed NFRecord.
	GetHorzResolution (see page 679)	Retrieves horizontal resolution in pixels per inch of fingerprint image from packed NFRecord.
	GetImpressionType (see page 680)	Retrieves impression type from packed NFRecord.
	GetMaxSize (see page 681)	Calculates the maximal NFRecord size.
	GetMaxSizeV1 (see page 681)	Calculates the maximal version 1.0 NFRecord size.
	GetPatternClass (see page 682)	Retrieves fingerprint pattern class from packed NFRecord.
	GetPosition (see page 683)	Retrieves finger position from packed NFRecord.
	GetQuality (see page 684)	Retrieves fingerprint quality from packed NFRecord
	GetSize (see page 684)	Calculates packed size of NFRecord.
	GetSizeV1 (see page 685)	Calculates version 1.0 packed size of NFRecord.
	GetVertResolution (see page 685)	Retrieves vertical resolution of fingerprint image from packed NFRecord.
	GetWidth (see page 686)	Retrieves width of fingerprint image from packed NFRecord.
	Save (see page 687)	Packs NFRecord to byte array.
	SaveV1 (see page 689)	Packs NFRecord to byte array in version 1.0 format.
	SortMinutiae (see page 691)	Sorts minutiae in NFRecord by the specified order.
	TruncateMinutiae (see page 691)	Truncates minutiae in NFRecord by peeling off the minutiae convex hull while minutia count is greater than the specified maximal count.
	TruncateMinutiaeByQuality (see page 691)	Truncates minutiae in NFRecord by removing minutiae which NMinutia.Quality (see page 694) field value is less than the specified threshold while minutia count is greater than the specified maximal count.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NFRecord Class

	Name	Description
	CbeffProductType (see page 691)	Gets or sets the Cbeff product type of the NFRecord.
	Cores (see page 692)	Gets cores collection.
	Deltas (see page 692)	Gets deltas collection.
	DoubleCores (see page 692)	Gets double cores collection.
	G (see page 692)	Gets or sets additional fingerprint coefficient.
	Height (see page 692)	Gets the height of fingerprint image.
	HorzResolution (see page 692)	Gets horizontal resolution of fingerprint image.
	ImpressionType (see page 693)	Gets or sets the impression type of the NFRecord.
	Minutiae (see page 693)	Gets minutiae collection.

	MinutiaeNeighbors (see page 693)	Gets minutia neighbors format from NFRecord.
	MinutiaFormat (see page 693)	Gets or sets minutia format from NFRecord.
	PatternClass (see page 693)	Gets or sets pattern class.
	Position (see page 694)	Gets or sets finger position.
	Quality (see page 694)	Gets or sets fingerprint quality.
	RidgeCountsType (see page 694)	Gets or sets ridge counts type.
	VertResolution (see page 694)	Gets vertical resolution of fingerprint image.
	Width (see page 694)	Gets width of fingerprint image.

6.2.2.1.8.1 NFRecord Constructor

6.2.2.1.8.1.1 NFRecord.NFRecord Constructor (IntPtr, int)

Initializes a new instance of the NFRecord class.

C#

```
public NFRecord(IntPtr pBuffer, int bufferLength);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord.
int bufferLength	Size of memory buffer that contains packed NFRecord.

6.2.2.1.8.1.2 NFRecord.NFRecord Constructor (IntPtr, int, out NFRecordInfo)

Initializes a new instance of the NFRecord class.

C#

```
public NFRecord(IntPtr pBuffer, int bufferLength, out NFRecordInfo info);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord.
int bufferLength	Size of memory buffer that contains packed NFRecord.
out NFRecordInfo info	The NFRecordInfo (see page 694) object.

6.2.2.1.8.1.3 NFRecord.NFRecord Constructor (IntPtr, int, uint)

Initializes a new instance of the NFRecord class.

C#

```
public NFRecord(IntPtr pBuffer, int bufferLength, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord.
int bufferLength	Size of memory buffer that contains packed NFRecord.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

Remarks

The following flags are supported: FlagSkipBlockedOrientations ([see page 674](#)), FlagSkipCurvatures ([see page 674](#)), FlagSkipGs ([see page 674](#)), FlagSkipQualities ([see page 674](#)), FlagSkipRidgeCounts ([see page 674](#)), FlagSkipSingularPoints ([see page 674](#)).

6.2.2.1.8.1.4 NfRecord.NfRecord Constructor (IntPtr, int, uint, out NfRecordInfo)

Initializes a new instance of the NfRecord class.

C#

```
public NfRecord(IntPtr pBuffer, int bufferLength, uint flags, out NfRecordInfo info);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NfRecord.
int bufferLength	Size of memory buffer that contains packed NfRecord.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
out NfRecordInfo info	The NfRecordInfo (see page 694) object.

Remarks

The following flags are supported: FlagSkipBlockedOrientations (see page 674), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipQualities (see page 674), FlagSkipRidgeCounts (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.1.5 NfRecord.NfRecord Constructor (bool, ushort, ushort, ushort, ushort)

Initializes a new instance of the NfRecord class.

C#

```
public NfRecord(bool isPalm, ushort width, ushort height, ushort horzResolution, ushort vertResolution);
```

Parameters

Parameters	Description
bool isPalm	Indicates that NfRecord was created for palmprints.
ushort width	Specifies width of fingerprint or palmprint image.
ushort height	Specifies height of fingerprint or palmprint image.
ushort horzResolution	Specifies horizontal resolution in pixels per inch of fingerprint or palmprint image.
ushort vertResolution	Specifies vertical resolution in pixels per inch of fingerprint or palmprint image.

6.2.2.1.8.1.6 NfRecord.NfRecord Constructor (bool, ushort, ushort, ushort, ushort, uint)

Initializes a new instance of the NfRecord class.

C#

```
public NfRecord(bool isPalm, ushort width, ushort height, ushort horzResolution, ushort vertResolution, uint flags);
```

Parameters

Parameters	Description
bool isPalm	Indicates if the NfRecord is for palmprints.
ushort width	Specifies width of fingerprint or palmprint image.
ushort height	Specifies height of fingerprint or palmprint image.
ushort horzResolution	Specifies horizontal resolution in pixels per inch of fingerprint or palmprint image.
ushort vertResolution	Specifies vertical resolution in pixels per inch of fingerprint or palmprint image.

uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

Remarks

The following flags are supported: FlagSkipBlockedOrient ([see page 674](#)), FlagSkipCurvatures ([see page 674](#)), FlagSkipGs ([see page 674](#)), FlagSkipQualities ([see page 674](#)), FlagSkipRidgeCounts ([see page 674](#)), FlagSkipSingularPoints ([see page 674](#)).

6.2.2.1.8.1.7 NFRecord.NFRecord Constructor (byte[])

Initializes a new instance of the NFRecord class.

C#

```
public NFRecord(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFRecord object.

6.2.2.1.8.1.8 NFRecord.NFRecord Constructor (byte[], out NFRecordInfo)

Initializes a new instance of the NFRecord class.

C#

```
public NFRecord(byte[] buffer, out NFRecordInfo info);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFRecord object.
out NFRecordInfo info	The NFRecordInfo (see page 694) object.

6.2.2.1.8.1.9 NFRecord.NFRecord Constructor (byte[], uint)

Initializes a new instance of the NFRecord class.

C#

```
public NFRecord(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFRecord object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

Remarks

The following flags are supported: FlagSkipBlockedOrient ([see page 674](#)), FlagSkipCurvatures ([see page 674](#)), FlagSkipGs ([see page 674](#)), FlagSkipQualities ([see page 674](#)), FlagSkipRidgeCounts ([see page 674](#)), FlagSkipSingularPoints ([see page 674](#)).

6.2.2.1.8.1.10 NFRecord.NFRecord Constructor (byte[], uint, out NFRecordInfo)

Initializes a new instance of the NFRecord class.

C#

```
public NFRecord(byte[] buffer, uint flags, out NFRecordInfo info);
```


Parameters

Parameters	Description
byte[] buffer	The packed NfRecord object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
out NfRecordInfo info	The NfRecordInfo (see page 694) object.

Remarks

The following flags are supported: FlagSkipBlockedOrientations (see page 674), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipQualities (see page 674), FlagSkipRidgeCounts (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.1.11 NfRecord.NfRecord Constructor (ushort, ushort, ushort, ushort)

Initializes a new instance of the NfRecord class.

C#

```
public NfRecord(ushort width, ushort height, ushort horzResolution, ushort vertResolution);
```

Parameters

Parameters	Description
ushort width	The fingerprint image width.
ushort height	The fingerprint image height.
ushort horzResolution	Horizontal resolution in pixels per inch of fingerprint image.
ushort vertResolution	Vertical resolution in pixels per inch of fingerprint image.

6.2.2.1.8.1.12 NfRecord.NfRecord Constructor (ushort, ushort, ushort, ushort, uint)

Initializes a new instance of the NfRecord class.

C#

```
public NfRecord(ushort width, ushort height, ushort horzResolution, ushort vertResolution, uint flags);
```

Parameters

Parameters	Description
ushort width	The fingerprint image width.
ushort height	The fingerprint image height.
ushort horzResolution	Horizontal resolution in pixels per inch of fingerprint image.
ushort vertResolution	Vertical resolution in pixels per inch of fingerprint image.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

Remarks

The following flags are supported: FlagSkipBlockedOrientations (see page 674), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipQualities (see page 674), FlagSkipRidgeCounts (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.2 NfRecord Classes

6.2.2.1.8.2.1 NfRecord.CoreCollection Class

Represents the collection of NfCore (see page 641).

Class Hierarchy**C#**

```
public sealed class CoreCollection : NStructCollection<NFCore>;
```

6.2.2.1.8.2.2 NRecord.DeltaCollection Class

Represents the collection of NFDelta ([see page 643](#)).

Class Hierarchy**C#**

```
public sealed class DeltaCollection : NStructCollection<NFDelta>;
```

6.2.2.1.8.2.3 NRecord.DoubleCoreCollection Class

Represents the collection of NFDoubleCore ([see page 647](#)).

Class Hierarchy**C#**

```
public sealed class DoubleCoreCollection : NStructCollection<NFDoubleCore>;
```

6.2.2.1.8.2.4 NRecord.MinutiaCollection Class

Represents the collection of NFMinutia ([see page 660](#)).

Class Hierarchy**C#**

```
public sealed class MinutiaCollection : NStructCollection<NFMinutia>;
```

6.2.2.1.8.2.5 NRecord.MinutiaNeighborsCollection Class

Represents the collection of NFMinutiaNeighbor ([see page 664](#)).

Class Hierarchy**C#**

```
public sealed class MinutiaNeighborsCollection : NStructArrayCollection<NFMinutiaNeighbor, NFMinutia>;
```

6.2.2.1.8.3 NRecord Fields**6.2.2.1.8.3.1 NRecord.FlagSaveBlockedOrientations Field**

The flag indicating whether blocked orientations should be packed in NRecord ([see page 666](#)).

C#

```
public const uint FlagSaveBlockedOrientations = 0x00040000;
```

6.2.2.1.8.3.2 NRecord.FlagSaveV2 Field

The flag indicating whether NRecord ([see page 666](#)) should be saved using version 2 algorithm.

C#

```
public const uint FlagSaveV2 = 0x20000000;
```

6.2.2.1.8.3.3 NRecord.FlagSaveV3 Field

The flag indicating whether NRecord ([see page 666](#)) should be saved using version 3 algorithm.

C#

```
public const uint FlagSaveV3 = 0x30000000;
```

6.2.2.1.8.3.4 NfRecord.FlagSkipBlockedOrientations Field

The flag indicating whether blocked orientations should be skipped while unpacking NfRecord (see page 666).

C#

```
public const uint FlagSkipBlockedOrientations = 0x00040000;
```

6.2.2.1.8.3.5 NfRecord.FlagSkipCurvatures Field

The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NfRecord (see page 666).

C#

```
public const uint FlagSkipCurvatures = 0x00200000;
```

6.2.2.1.8.3.6 NfRecord.FlagSkipGs Field

The flag indicating whether G (see page 692) field should be skipped while unpacking or packing NfRecord (see page 666).

C#

```
public const uint FlagSkipGs = 0x00400000;
```

6.2.2.1.8.3.7 NfRecord.FlagSkipQualities Field

The flag indicating whether minutiae qualities should be skipped while unpacking or packing NfRecord (see page 666).

C#

```
public const uint FlagSkipQualities = 0x00100000;
```

6.2.2.1.8.3.8 NfRecord.FlagSkipRidgeCounts Field

The flag indicating whether ridge counts should be skipped while unpacking or packing NfRecord (see page 666).

C#

```
public const uint FlagSkipRidgeCounts = 0x00010000;
```

6.2.2.1.8.3.9 NfRecord.FlagSkipSingularPoints Field

The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NfRecord (see page 666).

C#

```
public const uint FlagSkipSingularPoints = 0x00020000;
```

6.2.2.1.8.3.10 NfRecord.MaxCoreCount Field

The maximum number of cores a NfRecord (see page 666) can contain.

C#

```
[Obsolete("Use the MaxFingerCoreCount instead.")]  
public const int MaxCoreCount = MaxFingerCoreCount;
```

6.2.2.1.8.3.11 NfRecord.MaxDeltaCount Field

The maximum number of deltas a NfRecord (see page 666) can contain.

C#

```
[Obsolete("Use the MaxFingerDeltaCount instead.")]  
public const int MaxDeltaCount = MaxFingerDeltaCount;
```

6.2.2.1.8.3.12 NfRecord.MaxDimension Field

The maximum value for x and y coordinates of a minutia, core, delta or double core in a NfRecord (see page 666).

C#

```
[Obsolete("Use the MaxFingerDimension instead.")]  
public const ushort MaxDimension = MaxFingerDimension;
```

6.2.2.1.8.3.13 NfRecord.MaxDoubleCoreCount Field

The maximum number of double cores a NfRecord (see page 666) can contain.

C#

```
[Obsolete("Use the MaxFingerDoubleCoreCount instead.")]  
public const int MaxDoubleCoreCount = MaxFingerDoubleCoreCount;
```

6.2.2.1.8.3.14 NfRecord.MaxFingerCoreCount Field

The maximum number of cores a NfRecord (see page 666) can contain.

C#

```
public const int MaxFingerCoreCount = 15;
```

6.2.2.1.8.3.15 NfRecord.MaxFingerDeltaCount Field

The maximum number of deltas a NfRecord (see page 666) can contain.

C#

```
public const int MaxFingerDeltaCount = 15;
```

6.2.2.1.8.3.16 NfRecord.MaxFingerDimension Field

The maximum number of finger dimension.

C#

```
public const ushort MaxFingerDimension = 2047;
```

6.2.2.1.8.3.17 NfRecord.MaxFingerDoubleCoreCount Field

The maximum number of double cores a NfRecord (see page 666) can contain.

C#

```
public const int MaxFingerDoubleCoreCount = 15;
```

6.2.2.1.8.3.18 NfRecord.MaxFingerMinutiaCount Field

The maximum number of finger minutiae a NfRecord (see page 666) can contain.

C#

```
public const int MaxFingerMinutiaCount = 255;
```

6.2.2.1.8.3.19 NfRecord.MaxMinutiaCount Field

The maximum number of minutiae a NfRecord (see page 666) can contain.

C#

```
[Obsolete("Use the MaxFingerMinutiaCount instead.")]  
public const int MaxMinutiaCount = MaxFingerMinutiaCount;
```

6.2.2.1.8.3.20 NfRecord.MaxPalmCoreCount Field

The maximum number of cores in a palmprint a NfRecord (see page 666) can contain.

C#

```
public const int MaxPalmCoreCount = 255;
```

6.2.2.1.8.3.21 NfRecord.MaxPalmDeltaCount Field

The maximum number of deltas in a palmprint a NfRecord (see page 666) can contain.

C#

```
public const int MaxPalmDeltaCount = 255;
```

6.2.2.1.8.3.22 NfRecord.MaxPalmDimension Field

The maximum number of a palmprint dimension.

C#

```
public const ushort MaxPalmDimension = 16383;
```

6.2.2.1.8.3.23 NfRecord.MaxPalmDoubleCoreCount Field

The maximum number of double cores in a palmprint a NfRecord (see page 666) can contain.

C#

```
public const int MaxPalmDoubleCoreCount = 255;
```

6.2.2.1.8.3.24 NfRecord.MaxPalmMinutiaCount Field

The maximum number of minutiae in a palmprint a NfRecord (see page 666) can contain.

C#

```
public const int MaxPalmMinutiaCount = 65535;
```

6.2.2.1.8.3.25 NfRecord.Resolution Field

The resolution of minutiae, cores, deltas and double cores coordinates in a NfRecord (see page 666).

C#

```
public const ushort Resolution = 500;
```

6.2.2.1.8.4 NfRecord Methods**6.2.2.1.8.4.1 Check Method****6.2.2.1.8.4.1.1 NfRecord.Check Method (IntPtr, int)**

Checks if format of packed NfRecord (see page 666) is correct.

C#

```
public static void Check(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NfRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NfRecord (see page 666).

6.2.2.1.8.4.1.2 NfRecord.Check Method (byte[])

Checks if format of packed NfRecord (see page 666) is correct.

C#

```
public static void Check(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NfRecord (see page 666) object.

6.2.2.1.8.4.2 NfRecord.Clone Method

Creates NfRecord (see page 666) object from another NfRecord (see page 666) object.

C#

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

Returns

The new NfRecord (see page 666) object.

6.2.2.1.8.4.3 FromHandle Method**6.2.2.1.8.4.3.1 NfRecord.FromHandle Method (IntPtr)**

Creates NfRecord (see page 666) object from handle.

C#

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

Parameters

Parameters	Description
IntPtr handle	Handle (see page 623) to unmanaged NfRecord (see page 666) object.

Returns

NfRecord (see page 666) object.

6.2.2.1.8.4.3.2 NfRecord.FromHandle Method (IntPtr, bool)

Creates NfRecord (see page 666) object from handle.

C#

```
public static NfRecord FromHandle(IntPtr handle, bool ownsHandle);
```

Parameters

Parameters	Description
IntPtr handle	Handle (see page 623) to unmanaged NfRecord (see page 666) object.
bool ownsHandle	If set to true, on destruction of the NfRecord (see page 666) the NfRecord (see page 666) pointed to by the handle will also be destroyed.

Returns

NfRecord (see page 666) object.

6.2.2.1.8.4.4 GetCbeffProductType Method**6.2.2.1.8.4.4.1 NfRecord.GetCbeffProductType Method (IntPtr, int)**

Retrieves the Cbeff product type of the NfRecord (see page 666).

C#

```
public static ushort GetCbeffProductType(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

The Cbeff product type.

6.2.2.1.8.4.4.2 NFRecord.GetCbeffProductType Method (byte[])

Retrieves the Cbeff product type of the NFRecord (see page 666).

C#

```
public static ushort GetCbeffProductType(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFRecord (see page 666) object.

Returns

The Cbeff product type.

6.2.2.1.8.4.5 GetG Method**6.2.2.1.8.4.5.1 NFRecord.GetG Method (IntPtr, int)**

Retrieves G (see page 692) from packed NFRecord (see page 666).

C#

```
public static byte GetG(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

The G (see page 692) from packed NFRecord (see page 666). G (see page 692) - average fingerprint ridge density. It can have values from 0 to 255, so it occupies one byte.

6.2.2.1.8.4.5.2 NFRecord.GetG Method (byte[])

Retrieves G (see page 692) from packed NFRecord (see page 666).

C#

```
public static byte GetG(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFRecord (see page 666) object.

Returns

The G (see page 692) from packed NFRecord (see page 666). G (see page 692) - average fingerprint ridge density. It can have values from 0 to 255, so it occupies one byte.

6.2.2.1.8.4.6 GetHeight Method**6.2.2.1.8.4.6.1 NFRecord.GetHeight Method (IntPtr, int)**

Retrieves height of fingerprint image from packed NFRecord (see page 666).

C#

```
public static ushort GetHeight(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

The height of fingerprint image.

6.2.2.1.8.4.6.2 NFRecord.GetHeight Method (byte[])

Retrieves height of fingerprint image from packed NFRecord (see page 666).

C#

```
public static ushort GetHeight(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFRecord (see page 666) object.

Returns

The height of fingerprint image.

6.2.2.1.8.4.7 GetHorzResolution Method**6.2.2.1.8.4.7.1 NFRecord.GetHorzResolution Method (IntPtr, int)**

Retrieves horizontal resolution in pixels per inch of fingerprint image from packed NFRecord (see page 666).

C#

```
public static ushort GetHorzResolution(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).

int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

The horizontal resolution in pixels per inch of fingerprint image.

6.2.2.1.8.4.7.2 NFRecord.GetHorzResolution Method (byte[])

Retrieves horizontal resolution in pixels per inch of fingerprint image from packed NFRecord ([see page 666](#)).

C#

```
public static ushort GetHorzResolution(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The byte array of packed NFRecord (see page 666).

Returns

The horizontal resolution in pixels per inch of fingerprint image.

6.2.2.1.8.4.8 GetImpressionType Method**6.2.2.1.8.4.8.1 NFRecord.GetImpressionType Method (IntPtr, int)**

Retrieves impression type from packed NFRecord ([see page 666](#)).

C#

```
public static NFImpressionType GetImpressionType(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

A NFImpressionType ([see page 747](#)) enumeration member specifying impression type of fingerprint.

6.2.2.1.8.4.8.2 NFRecord.GetImpressionType Method (byte[])

Retrieves impression type from packed NFRecord ([see page 666](#)).

C#

```
public static NFImpressionType GetImpressionType(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The byte array of packed NFRecord (see page 666).

Returns

A NFImpressionType ([see page 747](#)) enumeration member specifying impression type of fingerprint.

6.2.2.1.8.4.9 GetMaxSize Method

6.2.2.1.8.4.9.1 NfRecord.GetMaxSize Method (NfMinutiaFormat, int, NfRidgeCountsType, int, int, int)

Calculates the maximal NfRecord (see page 666) size.

C#

```
public static int GetMaxSize(NfMinutiaFormat minutiaFormat, int minutiaCount,
    NfRidgeCountsType ridgeCountsType, int coreCount, int deltaCount, int doubleCoreCount);
```

Parameters

Parameters	Description
NfMinutiaFormat minutiaFormat	One of the NfMinutiaFormat (see page 748) values.
int minutiaCount	The minutiae count.
NfRidgeCountsType ridgeCountsType	One of the NfRidgeCountsType (see page 752) values.
int coreCount	The cores count.
int deltaCount	The deltas count.
int doubleCoreCount	The double cores count.

Returns

The maximal NfRecord (see page 666) size. The size depends on method parameters.

6.2.2.1.8.4.9.2 NfRecord.GetMaxSize Method (int, bool, NfMinutiaFormat, int, NfRidgeCountsType, int, int, int, int, int)

Calculates the maximal NfRecord (see page 666) size.

C#

```
public static int GetMaxSize(int version, bool isPalm, NfMinutiaFormat minutiaFormat, int
    minutiaCount, NfRidgeCountsType ridgeCountsType, int coreCount, int deltaCount, int
    doubleCoreCount, int boWidth, int boHeight);
```

Parameters

Parameters	Description
int version	Specifies algorithm version used by this function. Can have these values: 0, 1, 2, 3. The default value is 0 (in this case third version of algorithm will be used).
bool isPalm	Boolean value indicating the type of the NfRecord (see page 666). If set to true, NfRecord (see page 666) contains palmprint record.
NfMinutiaFormat minutiaFormat	One of the NfMinutiaFormat (see page 748) values.
int minutiaCount	The minutiae count
NfRidgeCountsType ridgeCountsType	One of the NfRidgeCountsType (see page 752) values.
int coreCount	The cores count.
int deltaCount	The deltas count.
int doubleCoreCount	The double cores count.
int boWidth	The width of blocked orientations.
int boHeight	The height of blocked orientations.

Returns

The maximal NfRecord (see page 666) size. The size depends on method parameters.

6.2.2.1.8.4.10 GetMaxSizeV1 Method**6.2.2.1.8.4.10.1 NfRecord.GetMaxSizeV1 Method (NfMinutiaFormat, int, int, int, int)**

Calculates the maximal version 1.0 NfRecord (see page 666) size.

C#

```
public static int GetMaxSizeV1(NFMinutiaFormat minutiaFormat, int minutiaCount, int coreCount, int deltaCount, int doubleCoreCount);
```

Parameters

Parameters	Description
NFMinutiaFormat minutiaFormat	One of the NFMinutiaFormat (see page 748) values.
int minutiaCount	The minutiae count.
int coreCount	The cores count.
int deltaCount	The deltas count.
int doubleCoreCount	The double cores count.

Returns

The maximal version 1.0 NFRecord (see page 666) size. The size depends on method parameters.

6.2.2.1.8.4.10.2 NFRecord.GetMaxSizeV1 Method (NFMinutiaFormat, int, int, int, int, int, int)

Calculates the maximal version 1.0 NFRecord (see page 666) size.

C#

```
public static int GetMaxSizeV1(NFMinutiaFormat minutiaFormat, int minutiaCount, int coreCount, int deltaCount, int doubleCoreCount, int boWidth, int boHeight);
```

Parameters

Parameters	Description
NFMinutiaFormat minutiaFormat	One of the NFMinutiaFormat (see page 748) values.
int minutiaCount	The minutiae count.
int coreCount	The cores count.
int deltaCount	The deltas count.
int doubleCoreCount	The double cores count.
int boWidth	For compatibility with VeriFinger.
int boHeight	For compatibility with VeriFinger.

Returns

The maximal version 1.0 NFRecord (see page 666) size. The size depends on method parameters.

6.2.2.1.8.4.11 GetPatternClass Method**6.2.2.1.8.4.11.1 NFRecord.GetPatternClass Method (IntPtr, int)**

Retrieves fingerprint pattern class from packed NFRecord (see page 666).

C#

```
public static NFPatternClass GetPatternClass(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

One of the NFPatternClass (see page 749) values.

Remarks

1. This function supports both NfRecord (see page 666) version 1.0 and 2.0 formats. Always returns NfPatternClass.Unknown for version 1.0 format.
2. This method returns NfPatternClass (see page 749) if NfPatternClass (see page 749) was set using PatternClass (see page 693).

6.2.2.1.8.4.11.2 NfRecord.GetPatternClass Method (byte[])

Retrieves fingerprint pattern class from packed NfRecord (see page 666).

C#

```
public static NfPatternClass GetPatternClass(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The byte array of packed NfRecord (see page 666).

Returns

One of the NfPatternClass (see page 749) values.

Remarks

1. This function supports both NfRecord (see page 666) version 1.0 and 2.0 formats. Always returns NfPatternClass.Unknown for version 1.0 format.
2. This method returns NfPatternClass (see page 749) if NfPatternClass (see page 749) was set using PatternClass (see page 693).
3. Classify can be used for classification.

6.2.2.1.8.4.12 GetPosition Method**6.2.2.1.8.4.12.1 NfRecord.GetPosition Method (IntPtr, int)**

Retrieves finger position from packed NfRecord (see page 666).

C#

```
public static NfPosition GetPosition(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NfRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NfRecord (see page 666).

Returns

One of the NfPosition (see page 750) values.

6.2.2.1.8.4.12.2 NfRecord.GetPosition Method (byte[])

Retrieves finger position from packed NfRecord (see page 666).

C#

```
public static NfPosition GetPosition(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The byte array of packed NFRecord (see page 666).

Returns

One of the NFPosition ([see page 750](#)) values.

6.2.2.1.8.4.13 GetQuality Method**6.2.2.1.8.4.13.1 NFRecord.GetQuality Method (IntPtr, int)**

Retrieves fingerprint quality from packed NFRecord ([see page 666](#))

C#

```
public static byte GetQuality(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

The value of fingerprint quality.

6.2.2.1.8.4.13.2 NFRecord.GetQuality Method (byte[])

Retrieves fingerprint quality from packed NFRecord ([see page 666](#)).

C#

```
public static byte GetQuality(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The byte array of packed NFRecord (see page 666).

Returns

The value of fingerprint quality.

6.2.2.1.8.4.14 GetSize Method**6.2.2.1.8.4.14.1 NFRecord.GetSize Method ()**

Calculates packed size of NFRecord ([see page 666](#)).

C#

```
public int GetSize();
```

Returns

The packed size of NFRecord ([see page 666](#)).

Remarks

The method calculates current (2.0) version packed size of NFRecord ([see page 666](#)).

6.2.2.1.8.4.14.2 NFRecord.GetSize Method (uint)

Calculates packed size of NFRecord (see page 666). Behavior is controlled through flags.

C#

```
public int GetSize(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

The packed size of NFRecord (see page 666).

Remarks

The method calculates current (2.0) version packed size of NFRecord (see page 666). For the list of flags that are supported see Save (see page 687) function.

6.2.2.1.8.4.15 GetSizeV1 Method

6.2.2.1.8.4.15.1 NFRecord.GetSizeV1 Method ()

Calculates version 1.0 packed size of NFRecord (see page 666).

C#

```
public int GetSizeV1();
```

Returns

The packed size of NFRecord (see page 666).

6.2.2.1.8.4.15.2 NFRecord.GetSizeV1 Method (uint)

Calculates version 1.0 packed size of NFRecord (see page 666). Behavior is controlled through flags.

C#

```
public int GetSizeV1(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

The packed size of NFRecord (see page 666).

Remarks

For the list of flags that are supported see SaveV1 function.

6.2.2.1.8.4.16 GetVertResolution Method

6.2.2.1.8.4.16.1 NFRecord.GetVertResolution Method (IntPtr, int)

Retrieves vertical resolution of fingerprint image from packed NFRecord (see page 666).

C#

```
public static ushort GetVertResolution(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

The vertical resolution in pixels per inch of fingerprint image.

6.2.2.1.8.4.16.2 NFRecord.GetVertResolution Method (byte[])

Retrieves vertical resolution of fingerprint image from packed NFRecord ([see page 666](#)).

C#

```
public static ushort GetVertResolution(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The byte array of packed NFRecord (see page 666).

Returns

The vertical resolution in pixels per inch of fingerprint image.

6.2.2.1.8.4.17 GetWidth Method**6.2.2.1.8.4.17.1 NFRecord.GetWidth Method (IntPtr, int)**

Retrieves width of fingerprint image from packed NFRecord ([see page 666](#)).

C#

```
public static ushort GetWidth(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFRecord (see page 666).
int bufferSize	Size of memory buffer that contains packed NFRecord (see page 666).

Returns

Pointer to memory buffer that contains packed NFRecord ([see page 666](#)).

6.2.2.1.8.4.17.2 NFRecord.GetWidth Method (byte[])

Retrieves width of fingerprint image from packed NFRecord ([see page 666](#)).

C#

```
public static ushort GetWidth(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The byte array of packed NFRecord (see page 666).

Returns

The width of fingerprint image.

6.2.2.1.8.4.18 Save Method**6.2.2.1.8.4.18.1 NfRecord.Save Method ()**

Packs NfRecord (see page 666) to byte array.

C#

```
public byte[] Save();
```

Returns

The array of packed NfRecord (see page 666).

Remarks

The method packs NfRecord (see page 666) in current (2.0) version format. Note that blocked orientations are not packed. To pack them use Save with flags or SaveV1 (see page 689) method.

6.2.2.1.8.4.18.2 NfRecord.Save Method (IntPtr, int)

Packs NfRecord (see page 666) to memory buffer.

C#

```
public int Save(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer for the packed NfRecord (see page 666) object.
int bufferSize	Size of memory buffer that contains packed NfRecord (see page 666).

Returns

The size of packed NfRecord (see page 666)

6.2.2.1.8.4.18.3 NfRecord.Save Method (IntPtr, int, uint)

Packs NfRecord (see page 666) to memory buffer.

C#

```
public int Save(IntPtr pBuffer, int bufferSize, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer for the packed NfRecord (see page 666) object.
int bufferSize	Size of memory buffer that contains packed NfRecord (see page 666).
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

The size of packed NfRecord (see page 666)

Remarks

The method packs NfRecord (see page 666) in current (2.0) version format. Note that blocked orientations are not packed by default. The following flags are supported: FlagSaveBlockedOrientations (see page 673), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipQualities (see page 674), FlagSkipRidgeCounts (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.4.18.4 NfRecord.Save Method (byte[])

Packs NfRecord (see page 666) to byte array.

C#

```
public int Save(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NfRecord (see page 666) object.

Returns

The size of packed NfRecord (see page 666).

Remarks

The method packs NfRecord (see page 666) in current (2.0) version format.

6.2.2.1.8.4.18.5 NfRecord.Save Method (byte[], uint)

Packs NfRecord (see page 666) to byte array. Behavior is controlled through flags.

C#

```
public int Save(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	The packed NfRecord (see page 666) object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

The size of packed NfRecord (see page 666).

Remarks

The method packs NfRecord (see page 666) in current (2.0) version format. Note that blocked orientations are not packed by default. The following flags are supported: FlagSaveBlockedOrientations (see page 673), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipQualities (see page 674), FlagSkipRidgeCounts (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.4.18.6 NfRecord.Save Method (uint)

Packs NfRecord (see page 666) to byte array. Behavior is controlled through flags.

C#

```
public byte[] Save(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

The byte array of packed NfRecord (see page 666).

Remarks

The method packs NfRecord (see page 666) in current (2.0) version format. Note that blocked orientations are not packed by default. The following flags are supported: FlagSaveBlockedOrientations (see page 673), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipQualities (see page 674), FlagSkipRidgeCounts (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.4.19 SaveV1 Method

6.2.2.1.8.4.19.1 NfRecord.SaveV1 Method ()

Packs NfRecord (see page 666) to byte array in version 1.0 format.

C#

```
public byte[] SaveV1();
```

Returns

The byte array of packed NfRecord (see page 666).

6.2.2.1.8.4.19.2 NfRecord.SaveV1 Method (IntPtr, int)

Packs NfRecord (see page 666) to memory buffer in version 1.0 format

C#

```
public int SaveV1(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer for the packed NfRecord (see page 666) object.
int bufferSize	Size of memory buffer that contains packed NfRecord (see page 666).

Returns

The size of packed NfRecord (see page 666).

6.2.2.1.8.4.19.3 NfRecord.SaveV1 Method (IntPtr, int, uint)

Packs NfRecord (see page 666) to memory buffer in version 1.0 format

C#

```
public int SaveV1(IntPtr pBuffer, int bufferSize, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer for the packed NfRecord (see page 666) object.
int bufferSize	Size of memory buffer that contains packed NfRecord (see page 666).
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

The size of packed NfRecord (see page 666).

Remarks

Note that blocked orientations are not packed by default. The following flags are supported: FlagSaveBlockedOrientations (see page 673), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.4.19.4 NFRecord.SaveV1 Method (byte[])

Packs NFRecord (see page 666) to byte array in version 1.0 format.

C#

```
public int SaveV1(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFRecord (see page 666) object.

Returns

The size of packed NFRecord (see page 666).

Remarks

Note that blocked orientations are not packed by default. The following flags are supported: FlagSaveBlockedOrientations (see page 673), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.4.19.5 NFRecord.SaveV1 Method (byte[], uint)

Packs NFRecord (see page 666) to memory buffer in version 1.0 format.

C#

```
public int SaveV1(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFRecord (see page 666) object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

The size of packed NFRecord (see page 666).

Remarks

Note that blocked orientations are not packed by default. The following flags are supported: FlagSaveBlockedOrientations (see page 673), FlagSkipCurvatures (see page 674), FlagSkipGs (see page 674), FlagSkipSingularPoints (see page 674).

6.2.2.1.8.4.19.6 NFRecord.SaveV1 Method (uint)

Packs NFRecord (see page 666) to memory buffer in version 1.0 format.

C#

```
public byte[] SaveV1(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

The byte array of packed NFRecord (see page 666).

Remarks

Note that blocked orientations are not packed by default. The following flags are supported: `FlagSaveBlockedOrientations` (see page 673), `FlagSkipCurvatures` (see page 674), `FlagSkipGs` (see page 674), `FlagSkipSingularPoints` (see page 674).

6.2.2.1.8.4.20 NfRecord.SortMinutiae Method

Sorts minutiae in `NfRecord` (see page 666) by the specified order.

C#

```
public void SortMinutiae(NfMinutiaeOrder order);
```

Parameters

Parameters	Description
<code>NfMinutiaeOrder order</code>	<code>NfMinutiaeOrder</code> (see page 749) value.

6.2.2.1.8.4.21 NfRecord.TruncateMinutiae Method

Truncates minutiae in `NfRecord` (see page 666) by peeling off the minutiae convex hull while minutiae count is greater than the specified maximal count.

C#

```
public void TruncateMinutiae(int maxCount);
```

Parameters

Parameters	Description
<code>int maxCount</code>	Maximal minutiae count to be present in the <code>NfRecord</code> (see page 666) after truncation.

6.2.2.1.8.4.22 NfRecord.TruncateMinutiaeByQuality Method

Truncates minutiae in `NfRecord` (see page 666) by removing minutiae which `NfMinutiae.Quality` (see page 694) field value is less than the specified threshold while minutiae count is greater than the specified maximal count.

C#

```
public void TruncateMinutiaeByQuality(byte threshold, int maxCount);
```

Parameters

Parameters	Description
<code>byte threshold</code>	Specifies minimal <code>NfMinutiae.Quality</code> (see page 694) field value of minutiae not to be removed.
<code>int maxCount</code>	Maximal minutiae count to be present in the <code>NfRecord</code> (see page 666) after truncation.

6.2.2.1.8.5 NfRecord Properties

6.2.2.1.8.5.1 NfRecord.CbeffProductType Property

Gets or sets the Cbeff product type of the `NfRecord` (see page 666).

C#

```
public ushort CbeffProductType;
```

Property value

The Cbeff product type.

6.2.2.1.8.5.2 NFRecord.Cores Property

Gets cores collection.

C#

```
public CoreCollection Cores;
```

Returns

A CoreCollection (see page 672) that contains cores.

6.2.2.1.8.5.3 NFRecord.Deltas Property

Gets deltas collection.

C#

```
public DeltaCollection Deltas;
```

Returns

A DeltaCollection (see page 673) that contains deltas.

6.2.2.1.8.5.4 NFRecord.DoubleCores Property

Gets double cores collection.

C#

```
public DoubleCoreCollection DoubleCores;
```

Returns

A DoubleCoreCollection (see page 673) that contains double cores.

6.2.2.1.8.5.5 NFRecord.G Property

Gets or sets additional fingerprint coefficient.

C#

```
public byte G;
```

Remarks

G - average fingerprint ridge density. It can have values from 0 to 255, so it occupies one byte.

Property value

Fingerprint coefficient.

6.2.2.1.8.5.6 NFRecord.Height Property

Gets the height of fingerprint image.

C#

```
public ushort Height;
```

Property value

Height of fingerprint image.

6.2.2.1.8.5.7 NFRecord.HorzResolution Property

Gets horizontal resolution of fingerprint image.

C#

```
public ushort HorzResolution;
```

Property value

Horizontal resolution in pixels per inch of fingerprint image.

6.2.2.1.8.5.8 NFRecord.ImpressionType Property

Gets or sets the impression type of the NFRecord ([see](#) page 666).

C#

```
public NFImpressionType ImpressionType;
```

Property value

One of the NFImpressionType ([see](#) page 747) values. The default is Neurotec.Biometrics.NFImpressionType.LiveScanPlain.

6.2.2.1.8.5.9 NFRecord.Minutiae Property

Gets minutiae collection.

C#

```
public MinutiaCollection Minutiae;
```

Returns

A MinutiaCollection ([see](#) page 673) that contains minutiae.

6.2.2.1.8.5.10 NFRecord.MinutiaeNeighbors Property

Gets minutia neighbors format from NFRecord ([see](#) page 666).

C#

```
public MinutiaNeighborsCollection MinutiaeNeighbors;
```

Property value

A MinutiaNeighborsCollection ([see](#) page 673) collection.

6.2.2.1.8.5.11 NFRecord.MinutiaFormat Property

Gets or sets minutia format minutia format from NFRecord ([see](#) page 666).

C#

```
public NFMinutiaFormat MinutiaFormat;
```

Property value

One of the NFMinutiaFormat ([see](#) page 748) values. The default is Neurotec.Biometrics.NFMinutiaFormat.HasCurvature.

6.2.2.1.8.5.12 NFRecord.PatternClass Property

Gets or sets pattern class.

C#

```
public NFPatternClass PatternClass;
```

Remarks

This property gets NFPatternClass ([see](#) page 749) if NFPatternClass ([see](#) page 749) was set before getting.

Property value

One of the `NFPatternClass` (see page 749) values. By default is `Neurotec.Biometrics.NFPatternClass.Unknown`.

6.2.2.1.8.5.13 NFRecord.Position Property

Gets or sets finger position.

C#

```
public NFPosition Position;
```

Property value

One of the `NFPosition` (see page 750) values. By default is `Neurotec.Biometrics.NFPosition.Unknown`.

6.2.2.1.8.5.14 NFRecord.Quality Property

Gets or sets fingerprint quality.

C#

```
public byte Quality;
```

Property value

Fingerprint quality.

6.2.2.1.8.5.15 NFRecord.RidgeCountsType Property

Gets or sets ridge counts type.

C#

```
public NFRidgeCountsType RidgeCountsType;
```

Property value

One of the `NFRidgeCountsType` (see page 752) values.

6.2.2.1.8.5.16 NFRecord.VertResolution Property

Gets vertical resolution of fingerprint image.

C#

```
public ushort VertResolution;
```

Property value

Vertical resolution in pixels per inch of fingerprint image.

6.2.2.1.8.5.17 NFRecord.Width Property

Gets width of fingerprint image.

C#

```
public ushort Width;
```

Property value

Width of fingerprint image.

6.2.2.1.9 NFRecordInfo Class

For internal use.

Class Hierarchy

C#

```
public class NFRecordInfo : NInfo;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NFRecordInfo Properties

NFRecordInfo Class

	Name	Description
	HeaderSize (see page 695)	For internal use only.
	MajorVersion (see page 695)	For internal use only.
	MinorVersion (see page 695)	For internal use only.
	Size (see page 695)	For internal use only.

6.2.2.1.9.1 NFRecordInfo Properties

6.2.2.1.9.1.1 NFRecordInfo.HeaderSize Property

For internal use only.

C#

```
public ushort HeaderSize;
```

6.2.2.1.9.1.2 NFRecordInfo.MajorVersion Property

For internal use only.

C#

```
public byte MajorVersion;
```

6.2.2.1.9.1.3 NFRecordInfo.MinorVersion Property

For internal use only.

C#

```
public byte MinorVersion;
```

6.2.2.1.9.1.4 NFRecordInfo.Size Property

For internal use only.

C#

```
public uint Size;
```

6.2.2.1.10 NFTemplate Class

Provides functionality for packing, unpacking and editing Neurotechnology Fingers Templates (NFTemplates).

Class Hierarchy

C#

```
public sealed class NFTemplate : NObject, ICloneable;
```


NFTemplate Classes**NFTemplate Class**

	Name	Description
	RecordCollection (see page 701)	Collection of NFRecords.

NFTemplate Fields**NFTemplate Class**

	Name	Description
	FlagProcessFirstRecordOnly (see page 703)	The flag indicating whether only the first finger record should be unpacked or packed while unpacking or packing NFTemplate.
	MaxRecordCount (see page 703)	The maximum number of finger records NFTemplate can contain.

Methods**NFTemplate Class**

	Name	Description
	NFTemplate (see page 697)	Initializes a new instance of the NFTemplate class.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NFTemplate Class

	Name	Description
	CalculateSize (see page 703)	Calculates NFTemplate size.
	Check (see page 703)	Checks if format of packed NFTemplate format is correct.
	Clone (see page 704)	Creates NFTemplate object from another NFTemplate object.
	FromHandle (see page 704)	Creates NFTemplate object from handle.
	GetRecordCount (see page 705)	Retrieves records count.
	GetSize (see page 705)	Calculates packed size of NFTemplate object.
	IsPalmTemplate (see page 706)	Checks if the NFTemplate contains packed palm records.
	Pack (see page 707)	Packs NFRecord (see page 666) objects into byte array.
	Save (see page 708)	Packs NFTemplate object to byte array.
	Unpack (see page 710)	Unpacks NFTemplate object from memory buffer.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).

	Owner (see page 623)	Gets owner of the object.

NFTemplate Class

	Name	Description
	IsPalm (see page 712)	Checks if the NFTemplate contains palm records.
	Records (see page 712)	Gets Records collection.

6.2.2.1.10.1 NFTemplate Constructor

6.2.2.1.10.1.1 NFTemplate.NFTemplate Constructor ()

Initializes a new instance of the NFTemplate class.

C#

```
public NFTemplate();
```

6.2.2.1.10.1.2 NFTemplate.NFTemplate Constructor (IntPtr, int)

Initializes a new instance of the NFTemplate class.

C#

```
public NFTemplate(IntPtr pBuffer, int bufferLength);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFTemplate.
int bufferLength	Size of memory buffer that contains packed NFTemplate.

6.2.2.1.10.1.3 NFTemplate.NFTemplate Constructor (IntPtr, int, out NFTemplateInfo)

Initializes a new instance of the NFTemplate class.

C#

```
public NFTemplate(IntPtr pBuffer, int bufferLength, out NFTemplateInfo info);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFTemplate.
int bufferLength	Size of memory buffer that contains packed NFTemplate.
out NFTemplateInfo info	The NFTemplateInfo (see page 712) object.

6.2.2.1.10.1.4 NFTemplate.NFTemplate Constructor (IntPtr, int, uint)

Initializes a new instance of the NFTemplate class.

C#

```
public NFTemplate(IntPtr pBuffer, int bufferLength, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFTemplate.
int bufferLength	Size of memory buffer that contains packed NFTemplate.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

Remarks

The following flags are supported:

- FlagProcessFirstRecordOnly (see page 703)
- FlagSkipBlockedOrients
- FlagSkipCurvatures
- FlagSkipGs
- FlagSkipQualities
- FlagSkipRidgeCounts
- FlagSkipSingularPoints

are applied to each finger record contained in the NFTemplate.

6.2.2.1.10.1.5 NFTemplate.NFTemplate Constructor (IntPtr, int, uint, out NFTemplateInfo)

Initializes a new instance of the NFTemplate class.

C#

```
public NFTemplate(IntPtr pBuffer, int bufferLength, uint flags, out NFTemplateInfo info);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFTemplate.
int bufferLength	Size of memory buffer that contains packed NFTemplate.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
out NFTemplateInfo info	The NFTemplateInfo (see page 712) object.

Remarks

The following flags are supported:

- FlagProcessFirstRecordOnly (see page 703),
- FlagSkipBlockedOrients
- FlagSkipCurvatures
- FlagSkipGs
- FlagSkipQualities
- FlagSkipRidgeCounts
- FlagSkipSingularPoints

are applied to each finger record contained in the NFTemplate.

6.2.2.1.10.1.6 NFTemplate.NFTemplate Constructor (bool)

Initializes a new instance of the NFTemplate class.

C#

```
public NFTemplate(bool isPalm);
```

Parameters

Parameters	Description
bool isPalm	Boolean value indicating the type of NFTemplate. If set to true, NFTemplate contains palmprint record. If set to false, NFTemplate contains fingerprint record.

6.2.2.1.10.1.7 NTemplate.NTemplate Constructor (bool, uint)

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(bool isPalm, uint flags);
```

Parameters

Parameters	Description
bool isPalm	Boolean value indicating the type of NTemplate. If set to true, NTemplate contains palmpoint record. If set to false, NTemplate contains fingerprint record.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor. This parameter is reserved, must be zero.

6.2.2.1.10.1.8 NTemplate.NTemplate Constructor (byte[])

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	A byte array with packed NTemplate objects.

6.2.2.1.10.1.9 NTemplate.NTemplate Constructor (byte[], out NTemplateInfo)

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(byte[] buffer, out NTemplateInfo info);
```

Parameters

Parameters	Description
byte[] buffer	A byte array with packed NTemplate objects.
out NTemplateInfo info	NTemplateInfo (see page 712) object.

6.2.2.1.10.1.10 NTemplate.NTemplate Constructor (byte[], uint)

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	A byte array with packed NTemplate objects.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

Remarks

The following flags are supported:

- FlagProcessFirstRecordOnly (see page 703)
- FlagSkipBlockedOrients

- FlagSkipCurvatures
- FlagSkipGs
- FlagSkipQualities
- FlagSkipRidgeCounts
- FlagSkipSingularPoints

are applied to each finger record contained in the NFTemplate.

6.2.2.1.10.1.11 NFTemplate.NFTemplate Constructor (byte[], uint, out NFTemplateInfo)

Initializes a new instance of the NFTemplate class.

C#

```
public NFTemplate(byte[] buffer, uint flags, out NFTemplateInfo info);
```

Parameters

Parameters	Description
byte[] buffer	A byte array with packed NFTemplate objects.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
out NFTemplateInfo info	NFTemplateInfo (see page 712) object.

Remarks

The following flags are supported:

- FlagProcessFirstRecordOnly (see page 703)
- FlagSkipBlockedOrientations
- FlagSkipCurvatures
- FlagSkipGs
- FlagSkipQualities
- FlagSkipRidgeCounts
- FlagSkipSingularPoints

are applied to each finger record contained in the NFTemplate.

6.2.2.1.10.1.12 NFTemplate.NFTemplate Constructor (uint)

Initializes a new instance of the NFTemplate class.

C#

```
public NFTemplate(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

Remarks

The following flags are supported:

- FlagProcessFirstRecordOnly (see page 703)
- FlagSkipBlockedOrientations
- FlagSkipCurvatures
- FlagSkipGs

- FlagSkipQualities
- FlagSkipRidgeCounts
- FlagSkipSingularPoints

are applied to each finger record contained in the NFTemplate.

6.2.2.1.10.2 NFTemplate Classes

6.2.2.1.10.2.1 NFTemplate.RecordCollection Class

Collection of NFRecords.

Class Hierarchy

C#

```
public sealed class RecordCollection : NObjectCollection<NFRecord>;
```

RecordCollection Methods

	Name	Description
	Add (see page 701)	Adds an object to the end of the RecordCollection.
	AddCopy (see page 702)	Copies NFTemplate (see page 695) object to the RecordCollection.

6.2.2.1.10.2.1.1 RecordCollection Methods

6.2.2.1.10.2.1.1.1 Add Method

6.2.2.1.10.2.1.1.1.1 NFTemplate.RecordCollection.Add Method (byte[])

Adds an object to the end of the RecordCollection ([see page 701](#)).

C#

```
public NFRecord Add(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFTemplate (see page 695) object.

Returns

Added to the collection NFTemplate ([see page 695](#)) object.

6.2.2.1.10.2.1.1.1.2 NFTemplate.RecordCollection.Add Method (byte[], uint)

Adds an object to the end of the RecordCollection ([see page 701](#)).

C#

```
public NFRecord Add(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	The packed NFTemplate (see page 695) object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. This parameter is reserved, must be zero.

Returns

Added to the collection NFTemplate ([see page 695](#)) object.

6.2.2.1.10.2.1.1.1.3 NFTemplate.RecordCollection.Add Method (ushort, ushort, ushort, ushort)

Adds an object to the end of the RecordCollection (see page 701).

C#

```
public NRecord Add(ushort width, ushort height, ushort horzResolution, ushort vertResolution);
```

Parameters

Parameters	Description
ushort width	width of image.
ushort height	height of image.
ushort horzResolution	horizontal resolution in pixels per inch of image.
ushort vertResolution	vertical resolution in pixels per inch of image.

Returns

Added to the collection NFTemplate (see page 695) object.

6.2.2.1.10.2.1.1.1.4 NFTemplate.RecordCollection.Add Method (ushort, ushort, ushort, ushort, uint)

Adds an object to the end of the RecordCollection (see page 701).

C#

```
public NRecord Add(ushort width, ushort height, ushort horzResolution, ushort vertResolution, uint flags);
```

Parameters

Parameters	Description
ushort width	width of image.
ushort height	height of image.
ushort horzResolution	horizontal resolution in pixels per inch of image.
ushort vertResolution	vertical resolution in pixels per inch of image.
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. This parameter is reserved, must be zero.

Returns

Added to the collection NFTemplate (see page 695) object.

6.2.2.1.10.2.1.1.2 NFTemplate.RecordCollection.AddCopy Method

Copies NFTemplate (see page 695) object to the RecordCollection (see page 701).

C#

```
public NRecord AddCopy(NRecord srcRecord);
```

Parameters

Parameters	Description
NRecord srcRecord	The NFTemplate (see page 695) object.

Returns

Copied to the collection NFTemplate (see page 695) object.

6.2.2.1.10.3 NFTemplate Fields

6.2.2.1.10.3.1 NFTemplate.FlagProcessFirstRecordOnly Field

The flag indicating whether only the first finger record should be unpacked or packed while unpacking or packing NFTemplate (see page 695).

C#

```
public const uint FlagProcessFirstRecordOnly = 0x00000100;
```

6.2.2.1.10.3.2 NFTemplate.MaxRecordCount Field

The maximum number of finger records NFTemplate (see page 695) can contain.

C#

```
public const int MaxRecordCount = 255;
```

6.2.2.1.10.4 NFTemplate Methods

6.2.2.1.10.4.1 CalculateSize Method

6.2.2.1.10.4.1.1 NFTemplate.CalculateSize Method (bool, int[])

Calculates NFTemplate (see page 695) size.

C#

```
public static int CalculateSize(bool isPalm, int[] recordSizes);
```

Parameters

Parameters	Description
bool isPalm	Indicates if the record contains palms.
int[] recordSizes	An array that contains NFRrecords sizes.

Returns

The value of NFTemplate (see page 695) size.

6.2.2.1.10.4.1.2 NFTemplate.CalculateSize Method (int[])

Calculates NFTemplate (see page 695) size.

C#

```
public static int CalculateSize(int[] recordSizes);
```

Parameters

Parameters	Description
int[] recordSizes	An array that contains NFRrecords sizes.

Returns

A size of records.

6.2.2.1.10.4.2 Check Method

6.2.2.1.10.4.2.1 NFTemplate.Check Method (IntPtr, int)

Checks if format of packed NFTemplate (see page 695) format is correct.

C#

```
public static void Check(IntPtr pBuffer, int bufferSize);
```


Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer with packed NFTemplate (see page 695).
int bufferSize	Size of memory buffer.

6.2.2.1.10.4.2.2 NFTemplate.Check Method (byte[])

Checks if format of packed NFTemplate (see page 695) format is correct.

C#

```
public static void Check(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	A byte array with packed NFTemplate (see page 695).

Returns

true if NFTemplate (see page 695) format is correct, false otherwise.

6.2.2.1.10.4.3 NFTemplate.Clone Method

Creates NFTemplate (see page 695) object from another NFTemplate (see page 695) object.

C#

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

Returns

A NFTemplate (see page 695) object.

6.2.2.1.10.4.4 FromHandle Method**6.2.2.1.10.4.4.1 NFTemplate.FromHandle Method (IntPtr)**

Creates NFTemplate (see page 695) object from handle.

C#

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

Parameters

Parameters	Description
IntPtr handle	Handle (see page 623) to unmanaged NFTemplate (see page 695) object.

Returns

A NFTemplate (see page 695) object.

6.2.2.1.10.4.4.2 NFTemplate.FromHandle Method (IntPtr, bool)

Creates NFTemplate (see page 695) object from handle.

C#

```
public static NFTemplate FromHandle(IntPtr handle, bool ownsHandle);
```

Parameters

Parameters	Description
IntPtr handle	Handle (see page 623) to unmanaged NFTemplate (see page 695) object. Handle (see page 623) to unmanaged NFTemplate (see page 695) object.
bool ownsHandle	If set to true, on destruction of the NFTemplate (see page 695) the NFTemplate (see page 695) pointed to by the handle will also be destroyed.

Returns

A NFTemplate (see page 695) object.

6.2.2.1.10.4.5 GetRecordCount Method**6.2.2.1.10.4.5.1 NFTemplate.GetRecordCount Method (IntPtr, int)**

Retrieves records count.

C#

```
public static int GetRecordCount(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer with NFTemplate (see page 695).
int bufferSize	Size of memory buffer.
buffer	A byte array with NFTemplate (see page 695).

Returns

Records (see page 712) count.

6.2.2.1.10.4.5.2 NFTemplate.GetRecordCount Method (byte[])

Retrieves records count.

C#

```
public static int GetRecordCount(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	A byte array with NFTemplate (see page 695).

Returns

Records (see page 712) count.

6.2.2.1.10.4.6 GetSize Method**6.2.2.1.10.4.6.1 NFTemplate.GetSize Method ()**

Calculates packed size of NFTemplate (see page 695) object.

C#

```
public int GetSize();
```

Returns

Size of NFTemplate (see page 695) object.

6.2.2.1.10.4.6.2 NFTemplate.GetSize Method (uint)

Calculates packed size of NFTemplate (see page 695) object. Behavior is controlled through flags.

C#

```
public int GetSize(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

Size of NFTemplate (see page 695) object.

Remarks

The following flags are supported:

- FlagProcessFirstRecordOnly (see page 703)

Flags supported by Neurotec.Biometrics.NFRecord.Save (see page 687) function are applied to each finger record contained in the NFTemplate (see page 695).

6.2.2.1.10.4.7 IsPalmTemplate Method

6.2.2.1.10.4.7.1 NFTemplate.IsPalmTemplate Method (IntPtr, int)

Checks if the NFTemplate (see page 695) contains packed palm records.

C#

```
public static bool IsPalmTemplate(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer that contains packed NFTemplate (see page 695).
int bufferSize	Size of memory buffer that contains packed NFTemplate (see page 695).

Returns

true if the NFTemplate (see page 695) contains palm records; otherwise, false.

6.2.2.1.10.4.7.2 NFTemplate.IsPalmTemplate Method (byte[])

Checks if the NFTemplate (see page 695) contains packed palm records.

C#

```
public static bool IsPalmTemplate(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NFTemplate (see page 695).

Returns

true if the NFTemplate (see page 695) contains palm records; otherwise, false.

6.2.2.1.10.4.8 Pack Method

6.2.2.1.10.4.8.1 NFTemplate.Pack Method (IntPtr[], int[])

Packs NFRecord (see page 666) objects into byte array.

C#

```
public static byte[] Pack(IntPtr[] pRecords, int[] recordSizes);
```

Parameters

Parameters	Description
IntPtr[] pRecords	Pointer to memory bufer containing separate NFRecords.
int[] recordSizes	Array of NFRecord (see page 666) sizes.

Returns

A byte array with packed NFTemplate (see page 695) objects.

6.2.2.1.10.4.8.2 NFTemplate.Pack Method (bool, IntPtr[], int[])

Packs NFRecord (see page 666) objects into byte array.

C#

```
public static byte[] Pack(bool isPalm, IntPtr[] pRecords, int[] recordSizes);
```

Parameters

Parameters	Description
bool isPalm	Indicates if the records contains palms.
IntPtr[] pRecords	Pointer to memory bufer containing separate NFRecords.
int[] recordSizes	Array of NFRecord (see page 666) sizes.

Returns

A byte array with packed NFTemplate (see page 695) objects.

6.2.2.1.10.4.8.3 NFTemplate.Pack Method (bool, params byte[])

Packs NFRecord (see page 666) objects into byte array.

C#

```
public static byte[] Pack(bool isPalm, params byte[] records);
```

Parameters

Parameters	Description
bool isPalm	Indicates if the records contains palms.
params byte[] records	A byte array of NFRecord (see page 666) objects byte arrays.

Returns

A byte array with packed NFTemplate (see page 695) objects.

6.2.2.1.10.4.8.4 NFTemplate.Pack Method (params byte[])

Packs NFRecord (see page 666) objects into byte array.

C#

```
public static byte[] Pack(params byte[] records);
```

Parameters

Parameters	Description
params byte[] records	A byte array of NFRecord (see page 666) objects byte arrays.

Returns

A byte array with packed NFTemplate (see page 695) objects.

6.2.2.1.10.4.9 Save Method**6.2.2.1.10.4.9.1 NFTemplate.Save Method ()**

Packs NFTemplate (see page 695) object to byte array.

C#

```
public byte[] Save();
```

Returns

A byte array with packed NFTemplate (see page 695) object.

6.2.2.1.10.4.9.2 NFTemplate.Save Method (IntPtr, int)

Packs NFTemplate (see page 695) object to memory buffer.

C#

```
public int Save(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer with packed NFTemplate (see page 695) objects.
int bufferSize	Size of memory buffer.

Returns

The size of packed NFTemplate (see page 695).

6.2.2.1.10.4.9.3 NFTemplate.Save Method (IntPtr, int, uint)

Packs NFTemplate (see page 695) object to memory buffer.

C#

```
public int Save(IntPtr pBuffer, int bufferSize, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer with packed NFTemplate (see page 695) objects.
int bufferSize	Size of memory buffer.
uint flags	Bitwise combination of zero or more flags that control behavior of the method.

Returns

The size of packed NFTemplate (see page 695).

Remarks

The following flags are supported: FlagProcessFirstRecordOnly (see page 703), Flags used in NFRecord (see page

666) Save methods are applied to each finger record contained in the `NFTemplate` (see page 695). Exclusive-OR is performed when few flags needed.

6.2.2.1.10.4.9.4 `NFTemplate.Save Method (byte[])`

Packs `NFTemplate` (see page 695) object to byte array.

C#

```
public int Save(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	A byte array with packed <code>NFTemplate</code> (see page 695) objects.

Returns

The size of packed `NFTemplate` (see page 695).

6.2.2.1.10.4.9.5 `NFTemplate.Save Method (byte[], uint)`

Packs `NFTemplate` (see page 695) object to byte array. Behavior is controlled through flags.

C#

```
public int Save(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	A byte array with packed <code>NFTemplate</code> (see page 695) objects.
uint flags	Bitwise combination of zero or more flags that control behavior of the method.

Returns

The size of packed `NFTemplate` (see page 695).

Remarks

The following flags are supported: `FlagProcessFirstRecordOnly` (see page 703), Flags used in `NFRecord` (see page 666) Save methods are applied to each finger record contained in the `NFTemplate` (see page 695). Exclusive-OR is performed when few flags needed.

6.2.2.1.10.4.9.6 `NFTemplate.Save Method (uint)`

Packs `NFTemplate` (see page 695) object to byte array. Behavior is controlled through flags.

C#

```
public byte[] Save(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that control behavior of the method.

Returns

A byte array with packed `NFTemplate` (see page 695) object.

Remarks

The following flags are supported: `FlagProcessFirstRecordOnly` (see page 703), Flags used in `NFRecord` (see page

666) Save methods are applied to each finger record contained in the NFTemplate (see page 695). Exclusive-OR is performed when few flags needed. .

6.2.2.1.10.4.10 Unpack Method

6.2.2.1.10.4.10.1 NFTemplate.Unpack Method (IntPtr, int, out bool, out byte, out byte, out uint, out byte, out IntPtr[], out int[])

Unpacks NFTemplate (see page 695) object from memory buffer.

C#

```
public static int Unpack(IntPtr pBuffer, int bufferSize, out bool isPalm, out byte
majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr[]
pRecords, out int[] recordSizes);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFTemplate (see page 695) object.
int bufferSize	Size of memory buffer that contains packed NFTemplate (see page 695).
out bool isPalm	Boolean value indicating the type of NFTemplate (see page 695). If set to true, NFTemplate (see page 695) contains palmprint record. If set to false, NFTemplate (see page 695) contains fingerprint record.
out byte majorVersion	Major version of NFTemplate (see page 695).
out byte minorVersion	Minor version of NFTemplate (see page 695).
out uint size	The size of buffer from header of packed NFTemplate (see page 695). This size is equal to bufferSize.
out byte headerSize	The size of packed NFTemplate (see page 695) header size.
out IntPtr[] pRecords	The array of NFRecords.
out int[] recordSizes	An array that contains NFRecord (see page 666) sizes.

Returns

The records count.

6.2.2.1.10.4.10.2 NFTemplate.Unpack Method (IntPtr, int, out bool, out byte, out byte, out uint, out byte, out byte[], out byte[])

Unpacks NFTemplate (see page 695) object from memory buffer.

C#

```
public static void Unpack(IntPtr pBuffer, int bufferSize, out bool isPalm, out byte
majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[]
records);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NFTemplate (see page 695) object.
int bufferSize	Size of memory buffer that contains packed NFTemplate (see page 695).

out bool isPalm	Boolean value indicating the type of NFTemplate (see page 695). If set to true, NFTemplate (see page 695) contains palmprint record. If set to false, NFTemplate (see page 695) contains fingerprint record.
out byte majorVersion	Major version of NFTemplate (see page 695).
out byte minorVersion	Minor version of NFTemplate (see page 695).
out uint size	The size of buffer from header of packed NFTemplate (see page 695). This size is equal to bufferSize.
out byte headerSize	The size of packed NFTemplate (see page 695) header size.
out byte[] records	A byte array that contains NFRecord (see page 666) objects bytes arrays.

6.2.2.1.10.4.10.3 NFTemplate.Unpack Method (byte[], out bool, out byte, out byte, out uint, out byte, out IntPtr[], out int[])

Unpacks NFTemplate (see page 695) object from byte array.

C#

```
public static int Unpack(byte[] buffer, out bool isPalm, out byte majorVersion, out byte
minorVersion, out uint size, out byte headerSize, out IntPtr[] pRecords, out int[]
recordSizes);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NFTemplate (see page 695) object.
out bool isPalm	Boolean value indicating the type of NFTemplate (see page 695). If set to true, NFTemplate (see page 695) contains palmprint record. If set to false, NFTemplate (see page 695) contains fingerprint record.
out byte majorVersion	Major version of NFTemplate (see page 695).
out byte minorVersion	Minor version of NFTemplate (see page 695).
out uint size	The size of buffer from header of packed NFTemplate (see page 695).
out byte headerSize	The size of packed NFTemplate (see page 695) header size.
out IntPtr[] pRecords	The array of NFRecords.
out int[] recordSizes	An array that contains NFRecord (see page 666) sizes.

Returns

The records count.

6.2.2.1.10.4.10.4 NFTemplate.Unpack Method (byte[], out bool, out byte, out byte, out uint, out byte, out byte[])

Unpacks NFTemplate (see page 695) object from byte array.

C#

```
public static void Unpack(byte[] buffer, out bool isPalm, out byte majorVersion, out byte
minorVersion, out uint size, out byte headerSize, out byte[] records);
```


Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NFTemplate (see page 695) object.
out bool isPalm	Boolean value indicating the type of NFTemplate (see page 695). If set to true, NFTemplate (see page 695) contains palmprint record. If set to false, NFTemplate (see page 695) contains fingerprint record.
out byte majorVersion	Major version of NFTemplate (see page 695).
out byte minorVersion	Minor version of NFTemplate (see page 695).
out uint size	The size of buffer from header of packed NFTemplate (see page 695).
out byte headerSize	The size of packed NFTemplate (see page 695) header size.
out byte[] records	A byte array that contains NFRecord (see page 666) objects bytes arrays.

6.2.2.1.10.5 NFTemplate Properties**6.2.2.1.10.5.1 NFTemplate.IsPalm Property**

Checks if the NFTemplate (see page 695) contains palm records.

C#

```
public bool IsPalm;
```

Property value

true if NFTemplate (see page 695) contains palm records; otherwise, false.

6.2.2.1.10.5.2 NFTemplate.Records Property

Gets Records collection.

C#

```
public RecordCollection Records;
```

Property value

A RecordCollection (see page 701) collection.

6.2.2.1.11 NFTemplateInfo Class**Class Hierarchy**

C#

```
public class NFTemplateInfo : NInfo;
```

Notes

This class is for internal use only.

NFTemplateInfo Classes**NFTemplateInfo Class**

	Name	Description
	RecordInfoCollection (see page 713)	For internal use only.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NTemplateInfo Properties

NTemplateInfo Class

	Name	Description
	HeaderSize (see page 713)	For internal use.
	MajorVersion (see page 713)	For internal use.
	MinorVersion (see page 713)	For internal use.
	RecordInfos (see page 713)	For internal use.
	Size (see page 713)	For internal use.

6.2.2.1.11.1 NTemplateInfo Classes

6.2.2.1.11.1.1 NTemplateInfo.RecordInfoCollection Class

For internal use only.

Class Hierarchy

C#

```
public sealed class RecordInfoCollection : ReadOnlyCollection<NFRecordInfo>;
```

6.2.2.1.11.2 NTemplateInfo Properties

6.2.2.1.11.2.1 NTemplateInfo.HeaderSize Property

For internal use.

C#

```
public byte HeaderSize;
```

6.2.2.1.11.2.2 NTemplateInfo.MajorVersion Property

For internal use.

C#

```
public byte MajorVersion;
```

6.2.2.1.11.2.3 NTemplateInfo.MinorVersion Property

For internal use.

C#

```
public byte MinorVersion;
```

6.2.2.1.11.2.4 NTemplateInfo.RecordInfos Property

For internal use.

C#

```
public RecordInfoCollection RecordInfos;
```

6.2.2.1.11.2.5 NTemplateInfo.Size Property

For internal use.

C#

```
public uint Size;
```

6.2.2.1.12 NleDetectionDetails Structure

Structure containing face detection results.

C#

```
[Serializable]
[StructLayout(LayoutKind.Sequential)]
public struct NleDetectionDetails {
}
```

NleDetectionDetails Properties

	Name	Description
	Eyes (see page 714)	Gets or sets Information of the found eyes.
	EyesAvailable (see page 714)	Gets or sets boolean value determining if eyes were found in the found face.
	Face (see page 714)	Gets or sets information of the found face.
	FaceAvailable (see page 714)	Gets or sets a boolean value determining if face was found.

6.2.2.1.12.1 NleDetectionDetails Properties

6.2.2.1.12.1.1 NleDetectionDetails.Eyes Property

Gets or sets Information of the found eyes.

C#

```
public NleEyes Eyes;
```

Property value

NleEyed object that contains information about eyes.

6.2.2.1.12.1.2 NleDetectionDetails.EyesAvailable Property

Gets or sets boolean value determining if eyes were found in the found face.

C#

```
public bool EyesAvailable;
```

6.2.2.1.12.1.3 NleDetectionDetails.Face Property

Gets or sets information of the found face.

C#

```
public NleFace Face;
```

6.2.2.1.12.1.4 NleDetectionDetails.FaceAvailable Property

Gets or sets a boolean value determining if face was found.

C#

```
public bool FaceAvailable;
```

6.2.2.1.13 NleEyes Structure

Structure defining information of an eye pair.

```
C#
[Serializable]
[StructLayout(LayoutKind.Sequential)]
public struct NleEyes {
}
```

NleEyes Properties

	Name	Description
	First (see page 715)	Gets or sets first eye coordinates.
	FirstConfidence (see page 715)	Gets or sets first eye coordinate confidence
	Second (see page 715)	Gets or sets second eye coordinates.
	SecondConfidence (see page 715)	Gets or sets a double value which represents second eye coordinates confidence.

6.2.2.1.13.1 NleEyes Properties

6.2.2.1.13.1.1 NleEyes.First Property

Gets or sets first eye coordinates.

```
C#
public Point First;
```

6.2.2.1.13.1.2 NleEyes.FirstConfidence Property

Gets or sets first eye coordinate confidence

```
C#
public double FirstConfidence;
```

Property value

Double value which represents first eye coordinates confidence.

6.2.2.1.13.1.3 NleEyes.Second Property

Gets or sets second eye coordinates.

```
C#
public Point Second;
```

Property value

Second eye coordinates.

6.2.2.1.13.1.4 NleEyes.SecondConfidence Property

Gets or sets a double value which represents second eye coordinates confidence.

```
C#
public double SecondConfidence;
```

Property value

Double value which represents second eye coordinates confidence.

6.2.2.1.14 NleFace Structure

Structure containing information of a face location in an image.

C#

```
[Serializable]
[StructLayout(LayoutKind.Sequential)]
public struct NleFace {
}
```

NleFace Properties

	Name	Description
	Confidence (see page 716)	Determines how confident the face region was found.
	Rectangle (see page 716)	Gets or sets information of the rectangle covering the face region.
	Rotation (see page 716)	Gets or sets information on face rotation in the image.

6.2.2.1.14.1 NleFace Properties**6.2.2.1.14.1.1 NleFace.Confidence Property**

Determines how confident the face region was found.

C#

```
public double Confidence;
```

6.2.2.1.14.1.2 NleFace.Rectangle Property

Gets or sets information of the rectangle covering the face region.

C#

```
public Rectangle Rectangle;
```

Property value

A set of four integers that represent the location and size of a rectangle.

6.2.2.1.14.1.3 NleFace.Rotation Property

Gets or sets information on face rotation in the image.

C#

```
public NleRotation Rotation;
```

Property value

A set of three shorts that represent rotation of the face in the image in degrees.

6.2.2.1.15 NleRotation Structure

Structure containing information of a face rotation in an image as a deviation from frontal face in degrees.

C#

```
[Serializable]
[StructLayout(LayoutKind.Sequential)]
public struct NleRotation {
}
```

NleRotation Properties

	Name	Description
	Yaw (see page 717)	Face yaw angle in degrees.
	Pitch (see page 717)	Face pitch angle in degrees.
	Roll (see page 717)	Face roll angle in degrees.

6.2.2.1.15.1 NleRotation Properties

6.2.2.1.15.1.1 NleRotation.Yaw Property

Face yaw angle in degrees.

C#

```
public short Yaw;
```

6.2.2.1.15.1.2 NleRotation.Pitch Property

Face pitch angle in degrees.

C#

```
public short Pitch;
```

6.2.2.1.15.1.3 NleRotation.Roll Property

Face roll angle in degrees.

C#

```
public short Roll;
```

6.2.2.1.16 NLExtractor Class

Provides functionality for extracting Neurotechnology Face Records (NLRecord) from face images.

Class Hierarchy

C#

```
public sealed class NLExtractor : NObject;
```

NLExtractor Fields

NLExtractor Class

	Name	Description
	ParameterFaceConfidenceThreshold ( see page 719)	Specifies the threshold which is considered when looking for faces in an image. For each face candidate confidence parameter is calculated. With higher threshold value faces are selected more strictly by the face detection method.
	ParameterFaceQualityThreshold ( see page 719)	Identifier specifying the threshold which is considered when extracting facial features from the image. With higher threshold better quality of face image is required to successfully extract facial features. The value of this parameter can be in range [0..255]. The default value is 128.
	ParameterFavorLargestFace ( see page 720)	If set to true the largest face found in an image will be used for facial feature extraction instead of the one with the highest face confidence score. The default value is true.
	ParameterLivenessThreshold ( see page 720)	This threshold checks if faces are extracted from a live image stream. The higher the value of this parameter, the more strictly the stream is checked if the face in it is real (for example if it's not forged by showing a photo of a person in front a camera). The value of this parameter can be in range [0..100]. The default value is 50.
	ParameterMaxIod ( see page 720)	Identifier specifying maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection methods.
	ParameterMaxRecordsPerTemplate ( see page 720)	Sets the maximum number of records an extraction method can return in one NLTemplate. The value of this parameter can be in range [2..16]. The default value is 5.

	ParameterMaxRollAngleDeviation (see page 720)	Identifier specifying the maximum roll angle deviation from frontal face in degrees. The value of this parameter can be in range [0..180]. The default value is 15.
	ParameterMinIod (see page 720)	Identifier specifying minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection methods.
	ParameterTemplateSize (see page 720)	Identifier specifying template size parameter. Parameter value can be one of the TemplateSize (see page 726) enumeration members.
	ParameterUseLivenessCheck (see page 721)	If set to true liveness check is performed while doing facial feature extraction from an image stream.

Methods

NLExtractor Class

	Name	Description
	NLExtractor (see page 719)	Initializes a new instance of NLExtractor class.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
 	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
 	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NLExtractor Class

	Name	Description
	Compress (see page 721)	Compresses the extracted template to a format suitable for storing the template.
	DetectFace (see page 721)	Method to detect a face in an image, that best fits the requirements for the face enrollment requirements.
	DetectFaces (see page 722)	Method to detect all faces in a given image.
	DetectFacialFeatures (see page 722)	Finds locations of facial feature points.
	Extract (see page 722)	Describes the extract method of the NLExtractor, which is used to extract face feature templates from the images.
	ExtractNext (see page 723)	Describes the extract method of the NLExtractor, which is used to extract face feature templates from the images.
	ExtractStart (see page 723)	Extraction from an image set initialization method.
	ExtractUsingDetails (see page 723)	Extracts feature template from the face with given detection details from an image.
	Generalize (see page 724)	Method to produce a generalized face features template from a given set of templates of the same person face.
 	GetStaticParameter (see page 724)	Method to get a static NLExtractor parameter value by using parameter id.
 	SetStaticParameter (see page 724)	Method to set a NLExtractor parameter value using parameter id.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NLExtractor Class

	Name	Description
	FaceConfidenceThreshold (see page 725)	Gets or sets the threshold which is considered when looking for faces in an image. For each face candidate confidence parameter is calculated. With higher threshold value faces are selected more strictly by the face detection method. Must be in range [0..100].
	FaceQualityThreshold (see page 725)	Gets or sets the threshold which is considered when extracting facial features from the image. With higher threshold better quality of face image is required to successfully extract facial features.
	FavorLargestFace (see page 725)	Gets or sets a value indicating if the largest face will be used for facial features extraction.
	LivenessThreshold (see page 725)	Gets or sets liveness threshold which checks if faces are extracted from a live image stream. The higher the value of this parameter, the more strictly the stream is checked if the face in it is real (for example if it's not forged by showing a photo of a person in front a camera).
	MaxIod (see page 725)	Gets or sets maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].
	MaxRecordsPerTemplate (see page 726)	Gets or sets the maximum number of records an extraction method can return in one NLTemplate.
	MaxRollAngleDeviation (see page 726)	Gets or sets maximum roll angle deviation from frontal face in degrees which is considered when looking for faces in an image. Must be in range [0..180].
	MinIod (see page 726)	Sets or gets minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection methods. Must be in range [10..10000].
	TemplateSize (see page 726)	Gets or sets template size.
	UseLivenessCheck (see page 726)	Sets or gets a boolean value indicating whether to use liveness check.

6.2.2.1.16.1 NLExtractor.NLExtractor Constructor

Initializes a new instance of NLExtractor class.

C#

```
public NLExtractor();
```

6.2.2.1.16.2 NLExtractor Fields**6.2.2.1.16.2.1 NLExtractor.ParameterFaceConfidenceThreshold Field**

Specifies the threshold which is considered when looking for faces in an image. For each face candidate confidence parameter is calculated. With higher threshold value faces are selected more strictly by the face detection method.

C#

```
public const ushort ParameterFaceConfidenceThreshold = 10103;
```

6.2.2.1.16.2.2 NLExtractor.ParameterFaceQualityThreshold Field

Identifier specifying the threshold which is considered when extracting facial features from the image. With higher threshold

better quality of face image is required to successfully extract facial features. The value of this parameter can be in range [0..255]. The default value is 128.

C#

```
public const ushort ParameterFaceQualityThreshold = 10350;
```

6.2.2.1.16.2.3 NLExtractor.ParameterFavorLargestFace Field

If set to true the largest face found in an image will be used for facial feature extraction instead of the one with the highest face confidence score. The default value is true.

C#

```
public const ushort ParameterFavorLargestFace = 10104;
```

6.2.2.1.16.2.4 NLExtractor.ParameterLivenessThreshold Field

This threshold checks if faces are extracted from a live image stream. The higher the value of this parameter, the more strictly the stream is checked if the face in it is real (for example if it's not forged by showing a photo of a person in front a camera). The value of this parameter can be in range [0..100]. The default value is 50.

C#

```
public const ushort ParameterLivenessThreshold = 10403;
```

6.2.2.1.16.2.5 NLExtractor.ParameterMaxIod Field

Identifier specifying maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection methods.

C#

```
public const ushort ParameterMaxIod = 10102;
```

6.2.2.1.16.2.6 NLExtractor.ParameterMaxRecordsPerTemplate Field

Sets the maximum number of records an extraction method can return in one NLTemplate. The value of this parameter can be in range [2..16]. The default value is 5.

C#

```
public const ushort ParameterMaxRecordsPerTemplate = 10408;
```

6.2.2.1.16.2.7 NLExtractor.ParameterMaxRollAngleDeviation Field

Identifier specifying the maximum roll angle deviation from frontal face in degrees. The value of this parameter can be in range [0..180]. The default value is 15.

C#

```
public const ushort ParameterMaxRollAngleDeviation = 10105;
```

6.2.2.1.16.2.8 NLExtractor.ParameterMinIod Field

Identifier specifying minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection methods.

C#

```
public const ushort ParameterMinIod = 10101;
```

6.2.2.1.16.2.9 NLExtractor.ParameterTemplateSize Field

Identifier specifying template size parameter. Parameter value can be one of the TemplateSize (see page 726) enumeration members.

C#

```
public const ushort ParameterTemplateSize = 10360;
```

6.2.2.1.16.2.10 NLExtractor.ParameterUseLivenessCheck Field

If set to true liveness check is performed while doing facial feature extraction from an image stream.

C#

```
public const ushort ParameterUseLivenessCheck = 10402;
```

6.2.2.1.16.3 NLExtractor Methods

6.2.2.1.16.3.1 Compress Method

6.2.2.1.16.3.1.1 NLExtractor.Compress Method (NLTemplate)

Compresses the extracted template to a format suitable for storing the template.

C#

```
[Obsolete("Use the Compress(NLTemplate, NleTemplateSize) instead.")]  
public NLTemplate Compress(NLTemplate template);
```

Parameters

Parameters	Description
NLTemplate template	Byte array with face feature template to be compressed.

Returns

Compressed face feature template.

Notes

Method is obsolete, use the Compress(NLTemplate, NleTemplateSize (see page 753)) instead.

6.2.2.1.16.3.1.2 NLExtractor.Compress Method (NLTemplate, NleTemplateSize)

Compresses the extracted template to a format suitable for storing the template.

C#

```
public static NLTemplate Compress(NLTemplate template, NleTemplateSize  
compressedTemplateSize);
```

Parameters

Parameters	Description
NLTemplate template	Face feature template to be compressed.
NleTemplateSize compressedTemplateSize	Compressed template size.

Returns

Compressed face feature template.

6.2.2.1.16.3.2 NLExtractor.DetectFace Method

Method to detect a face in an image, that best fits the requirements for the face enrollment requirements.

C#

```
public bool DetectFace(NGrayscaleImage image, out NleFace face);
```

Parameters

Parameters	Description
NGrayscaleImage image	Source image, in which to search for a face.

out NleFace face	NleFace (see page 715) structure with information about the found face.

Returns

A boolean value indicating whether a face was found in the image.

Return Values

Return Values	Description
True	Returns true if face was found in the image.
False	Returns false if face was found in the image.

6.2.2.1.16.3.3 NLExtractor.DetectFaces Method

Method to detect all faces in a given image.

C#

```
public NleFace[] DetectFaces(NGrayscaleImage image);
```

Parameters

Parameters	Description
NGrayscaleImage image	Source image, in which to search for the faces.

Returns

Returns an array of NleFace (see page 715) structures with information of faces found in the image. If no faces were found returns NULL (see page 133).

Return Values

Return Values	Description
Array	Returns an array of NleFace (see page 715) structures with information of faces found in the image.
NULL (see page 133)	If no faces were found returns NULL (see page 133).

6.2.2.1.16.3.4 NLExtractor.DetectFacialFeatures Method

Finds locations of facial feature points.

C#

```
public NleDetectionDetails DetectFacialFeatures(NGrayscaleImage image, NleFace face);
```

Parameters

Parameters	Description
NGrayscaleImage image	Source image, in which to search for the faces.
NleFace face	NleFace (see page 715) structure with information about the face in the image.

Returns

Face detection routine results.

6.2.2.1.16.3.5 NLExtractor.Extract Method

Describes the extract method of the NLExtractor (see page 717), which is used to extract face feature templates from the images.

C#

```
public NLTemplate Extract(NGrayscaleImage image, out NleDetectionDetails detectionDetails, out NleExtractionStatus extractionStatus);
```

Parameters

Parameters	Description
NGrayscaleImage image	Source image, from which the face features should be extracted.
out NleDetectionDetails detectionDetails	Returns face detection routine results.
out NleExtractionStatus extractionStatus	One of the value.

Returns

NLTemplate object.

6.2.2.1.16.3.6 NLExtractor.ExtractNext Method

Describes the extract method of the NLExtractor (see page 717), which is used to extract face feature templates from the images.

C#

```
[Obsolete("Use ExtractNext(NGrayscaleImage, NleDetectionDetails, int, NLTemplate) instead")]  
public NleExtractionStatus ExtractNext(NGrayscaleImage image, out NleDetectionDetails  
detectionDetails, out NLTemplate template);
```

Parameters

Parameters	Description
NGrayscaleImage image	Source image, from which the face features should be extracted.
out NleDetectionDetails detectionDetails	Returns face detection routine results.
out NLTemplate template	NLTemplate object.

Returns

One of the NleExtractionStatus (see page 752) value.

Remarks

To check if the face was found, the template parameter should be checked - if its TRUE, the face was found.

6.2.2.1.16.3.7 NLExtractor.ExtractStart Method

Extraction from an image set initialization method.

C#

```
public void ExtractStart(int durationInFrames);
```

Parameters

Parameters	Description
int durationInFrames	Sets the number of frames on which face detection should be performed.

6.2.2.1.16.3.8 NLExtractor.ExtractUsingDetails Method

Extracts feature template from the face with given detection details from an image.

C#

```
public NLTemplate ExtractUsingDetails(NGrayscaleImage image, NleDetectionDetails  
detectionDetails, out NleExtractionStatus extractionStatus);
```

Parameters

Parameters	Description
NGrayscaleImage image	Source image.

NleDetectionDetails detectionDetails	Face detection results.
out NleExtractionStatus extractionStatus	One of the NleExtractionStatus (see page 752) value.

Returns

NLTemplate object.

6.2.2.1.16.3.9 NLExtractor.Generalize Method

Method to produce a generalized face features template from a given set of templates of the same person face.

C#

```
[Obsolete("Use Generalize(NLTemplate[], NleExtractionStatus, int) instead.")]  
public NLTemplate Generalize(NLTemplate[] templates);
```

Parameters

Parameters	Description
NLTemplate[] templates	Array of templates to be generalized.

Returns

NLTemplate object.

6.2.2.1.16.3.10 NLExtractor.GetStaticParameter Method

Method to get a static NLExtractor (see page 717) parameter value by using parameter id.

C#

```
[Obsolete("Use NativeType.GetParameter instead.")]  
public static object GetStaticParameter(ushort parameterId);
```

Parameters

Parameters	Description
ushort parameterId	The parameter identifier.

Returns

The parameter value.

Remarks

Unused method.

6.2.2.1.16.3.11 NLExtractor.SetStaticParameter Method

Method to set a NLExtractor (see page 717) parameter value using parameter id.

C#

```
[Obsolete("Use NativeType.SetParameter instead.")]  
public static void SetStaticParameter(ushort parameterId, object value);
```

Parameters

Parameters	Description
ushort parameterId	The parameter identifier.
object value	The parameter value.

Remarks

Unused method.

6.2.2.1.16.4 NLExtractor Properties

6.2.2.1.16.4.1 NLExtractor.FaceConfidenceThreshold Property

Gets or sets the threshold which is considered when looking for faces in an image. For each face candidate confidence parameter is calculated. With higher threshold value faces are selected more strictly by the face detection method. Must be in range [0..100].

C#

```
public double FaceConfidenceThreshold;
```

Property value

Double value which represents face confidence threshold value.

6.2.2.1.16.4.2 NLExtractor.FaceQualityThreshold Property

Gets or sets the threshold which is considered when extracting facial features from the image. With higher threshold better quality of face image is required to successfully extract facial features.

C#

```
public byte FaceQualityThreshold;
```

Property value

Byte value which represents face quality threshold.

6.2.2.1.16.4.3 NLExtractor.FavorLargestFace Property

Gets or sets a value indicating if the largest face will be used for facial features extraction.

C#

```
public bool FavorLargestFace;
```

Property value

Boolean value indicating if the largest face should be used for facial features extraction. If set to true the largest face found in an image will be used for facial feature extraction instead of the one with the highest ParameterFaceConfidenceThreshold (see page 719) score.

6.2.2.1.16.4.4 NLExtractor.LivenessThreshold Property

Gets or sets liveness threshold which checks if faces are extracted from a live image stream. The higher the value of this parameter, the more strictly the stream is checked if the face in it is real (for example if it's not forged by showing a photo of a person in front a camera).

C#

```
public double LivenessThreshold;
```

Notes

The value can be in range [0..100]. The default value is 50.

Property value

Liveness threshold value.

6.2.2.1.16.4.5 NLExtractor.MaxIod Property

Gets or sets maximum distance between eyes in face. Faces which have greater distance between eyes than this parameter, won't be returned by the face detection routines. Must be in range [10..10000].

C#

```
public int MaxIod;
```

Property value

Integer value which represents maximum lod.

6.2.2.1.16.4.6 NLExtractor.MaxRecordsPerTemplate Property

Gets or sets the maximum number of records an extraction method can return in one NLTemplate.

C#

```
public int MaxRecordsPerTemplate;
```

Property value

Integer value which represents maximum number of record per template. The value can be in range [2..16]. The default value is 5.

6.2.2.1.16.4.7 NLExtractor.MaxRollAngleDeviation Property

Gets or sets maximum roll angle deviation from frontal face in degrees which is considered when looking for faces in an image. Must be in range [0..180].

C#

```
public short MaxRollAngleDeviation;
```

Property value

Short value which represents maximum face roll angle deviation value.

6.2.2.1.16.4.8 NLExtractor.MinIod Property

Sets or gets minimum distance between eyes in face. Faces which have smaller distance between eyes than this parameter, won't be returned by the face detection methods. Must be in range [10..10000].

C#

```
public int MinIod;
```

6.2.2.1.16.4.9 NLExtractor.TemplateSize Property

Gets or sets template size.

C#

```
public NleTemplateSize TemplateSize;
```

Property value

One of the NleTemplateSize (see page 753) value.

6.2.2.1.16.4.10 NLExtractor.UseLivenessCheck Property

Sets or gets a boolean value indicating whether to use liveness check.

C#

```
public bool UseLivenessCheck;
```

Property value

Boolean value indicating whether to use liveness check. The property value is true if liveness check is performed while doing facial feature extraction from an image stream. Otherwise, the property value is false.

6.2.2.1.17 NTemplate Class

Provides functionality for packing, unpacking and editing Neurotechnology Templates (NTemplates (see page 605)).

Class Hierarchy

C#

```
public sealed class NTemplate : NObject, ICloneable;
```

Methods

NTemplate Class

	Name	Description
	NTemplate (see page 728)	Initializes a new instance of the NTemplate class.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NTemplate Class

	Name	Description
	AddFaces (see page 730)	Creates new NLTemplate object.
	AddFacesCopy (see page 731)	Creates one NLTemplate object from another.
	AddFingers (see page 732)	Creates new NFTemplate (see page 695) object.
	AddFingersCopy (see page 733)	Creates one NFTemplate (see page 695) object from another.
	AddIrises (see page 733)	Creates new NETemplate object.
	AddIrisesCopy (see page 734)	Creates one NETemplate object from another.
	AddPalms (see page 734)	Creates new NFTemplate (see page 695) object.
	AddPalmsCopy (see page 735)	Creates one NFTemplate (see page 695) object from another.
	CalculateSize (see page 735)	Calculates the size of a packed NTemplate containing fingers, faces, irises and palms templates of the specified size.
	Check (see page 736)	Checks if format of packed NTemplate format is correct.
	Clear (see page 736)	Removes all NFRecord (see page 666) objects.
	Clone (see page 736)	Creates NTemplate object from another NTemplate object.
	FromHandle (see page 737)	Creates NTemplate object from handle.
	GetSize (see page 737)	Calculates packed size of NFTemplate (see page 695) object.
	Pack (see page 738)	Packs packed fingers template as NTemplate into the specified memory buffer.
	RemoveFaces (see page 739)	Removes faces template from the NTemplate.
	RemoveFingers (see page 739)	Removes fingers template from the NTemplate.
	RemoveIrises (see page 739)	Removes irises template from the NTemplate.
	RemovePalms (see page 739)	Removes palms template from the NTemplate.
	Save (see page 739)	Packs NTemplate object to byte array.
	Unpack (see page 741)	Unpacks packed fingers template from the packed NTemplate.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NTemplate Class

	Name	Description
	Faces (see page 743)	Gets NTemplate object.
	Fingers (see page 743)	Gets NTemplate (see page 695) object.
	Irises (see page 744)	Gets NTemplate object.
	Palms (see page 744)	Gets NTemplate (see page 695) object.

6.2.2.1.17.1 NTemplate Constructor**6.2.2.1.17.1.1 NTemplate.NTemplate Constructor ()**

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate();
```

6.2.2.1.17.1.2 NTemplate.NTemplate Constructor (IntPtr, int)

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(IntPtr pBuffer, int bufferLength);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NTemplate.
int bufferLength	Size of memory buffer that contains packed NTemplate.

6.2.2.1.17.1.3 NTemplate.NTemplate Constructor (IntPtr, int, out NTemplateInfo)

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(IntPtr pBuffer, int bufferLength, out NTemplateInfo info);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NTemplate.
int bufferLength	Size of memory buffer that contains packed NTemplate.
out NTemplateInfo info	The NTemplateInfo (see page 744) object.

6.2.2.1.17.1.4 NTemplate.NTemplate Constructor (IntPtr, int, uint)

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(IntPtr pBuffer, int bufferLength, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NTemplate.
int bufferLength	Size of memory buffer that contains packed NTemplate.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor. This parameter is reserved, must be zero.

6.2.2.1.17.1.5 NTemplate.NTemplate Constructor (IntPtr, int, uint, out NTemplateInfo)

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(IntPtr pBuffer, int bufferLength, uint flags, out NTemplateInfo info);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NTemplate.
int bufferLength	Size of memory buffer that contains packed NTemplate.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor. This parameter is reserved, must be zero.
out NTemplateInfo info	The NTemplateInfo (see page 744) object.

6.2.2.1.17.1.6 NTemplate.NTemplate Constructor (byte[])

Initializes a new instance of the NTemplate class from byte array.

C#

```
public NTemplate(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NTemplate.

6.2.2.1.17.1.7 NTemplate.NTemplate Constructor (byte[], out NTemplateInfo)

Initializes a new instance of the NTemplate class from byte array.

C#

```
public NTemplate(byte[] buffer, out NTemplateInfo info);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NTemplate.
out NTemplateInfo info	The NTemplateInfo (see page 744) object.

6.2.2.1.17.1.8 NTemplate.NTemplate Constructor (byte[], uint)

Initializes a new instance of the NTemplate class from byte array. Behavior is controlled through flags.

C#

```
public NTemplate(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NTemplate.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor. This parameter is reserved, must be zero.

6.2.2.1.17.1.9 NTemplate.NTemplate Constructor (byte[], uint, out NTemplateInfo)

Initializes a new instance of the NTemplate class from byte array. Behavior is controlled through flags.

C#

```
public NTemplate(byte[] buffer, uint flags, out NTemplateInfo info);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NTemplate.
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor. This parameter is reserved, must be zero.
out NTemplateInfo info	The FMRecordInfo object.

6.2.2.1.17.1.10 NTemplate.NTemplate Constructor (uint)

Initializes a new instance of the NTemplate class.

C#

```
public NTemplate(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the constructor. This parameter is reserved, must be zero.

6.2.2.1.17.2 NTemplate Methods**6.2.2.1.17.2.1 AddFaces Method****6.2.2.1.17.2.1.1 NTemplate.AddFaces Method ()**

Creates new NLTemplate object.

C#

```
public NLTemplate AddFaces();
```

Returns

A NLTemplate object.

6.2.2.1.17.2.1.2 NTemplate.AddFaces Method (byte[])

Creates new NLTemplate object.

C#

```
public NLTemplate AddFaces(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NLTemplate object.

Returns

A NLTemplate object.

6.2.2.1.17.2.1.3 NLTemplate.AddFaces Method (byte[], uint)

Creates new NLTemplate object.

C#

```
public NLTemplate AddFaces(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	The packed NLTemplate object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. Must be zero.

Returns

A NLTemplate object.

6.2.2.1.17.2.1.4 NLTemplate.AddFaces Method (uint)

Creates new NLTemplate object.

C#

```
public NLTemplate AddFaces(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. Must be zero.

Returns

A NLTemplate object.

6.2.2.1.17.2.2 NLTemplate.AddFacesCopy Method

Creates one NLTemplate object from another.

C#

```
public NLTemplate AddFacesCopy(NLTemplate srcFaces);
```

Parameters

Parameters	Description
NLTemplate srcFaces	The packed NLTemplate object.

Returns

A NLTemplate object.

6.2.2.1.17.2.3 AddFingers Method

6.2.2.1.17.2.3.1 NTemplate.AddFingers Method ()

Creates new NTemplate (see page 695) object.

C#

```
public NTemplate AddFingers();
```

Returns

A NTemplate (see page 695) object.

6.2.2.1.17.2.3.2 NTemplate.AddFingers Method (byte[])

Creates new NTemplate (see page 695) object from packed NTemplate (see page 695) object.

C#

```
public NTemplate AddFingers(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NTemplate (see page 695) object.

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.3.3 NTemplate.AddFingers Method (byte[], uint)

Creates new NTemplate (see page 695) object from packed NTemplate (see page 695) object.

C#

```
public NTemplate AddFingers(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	The packed NTemplate (see page 695) object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. Must be zero.

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.3.4 NTemplate.AddFingers Method (uint)

Creates new NTemplate (see page 695) object.

C#

```
public NTemplate AddFingers(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. Must be zero.

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.4 NTemplate.AddFingersCopy Method

Creates one NTemplate (see page 695) object from another.

C#

```
public NTemplate AddFingersCopy(NTemplate srcFingers);
```

Parameters

Parameters	Description
NTemplate srcFingers	The NTemplate (see page 695) object.

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.5 AddIris Method

6.2.2.1.17.2.5.1 NTemplate.AddIris Method ()

Creates new NTemplate object.

C#

```
public NTemplate AddIris();
```

Returns

The NTemplate object.

6.2.2.1.17.2.5.2 NTemplate.AddIris Method (byte[])

Creates new NTemplate object.

C#

```
public NTemplate AddIris(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NTemplate object.

Returns

The NTemplate object.

6.2.2.1.17.2.5.3 NTemplate.AddIris Method (byte[], uint)

Creates new NTemplate object.

C#

```
public NTemplate AddIris(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	The packed NTemplate object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. Must be zero.

Returns

The NTemplate object.

6.2.2.1.17.2.5.4 NTemplate.AddIrises Method (uint)

Creates new NTemplate object.

C#

```
public NTemplate AddIrises(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. Must be zero.

Returns

The NTemplate object.

6.2.2.1.17.2.6 NTemplate.AddIrisesCopy Method

Creates one NTemplate object from another.

C#

```
public NTemplate AddIrisesCopy(NTemplate srcIrises);
```

Parameters

Parameters	Description
NTemplate srcIrises	The NTemplate object.

Returns

The NTemplate object.

6.2.2.1.17.2.7 AddPalms Method

6.2.2.1.17.2.7.1 NTemplate.AddPalms Method ()

Creates new NTemplate (see page 695) object.

C#

```
public NTemplate AddPalms();
```

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.7.2 NTemplate.AddPalms Method (byte[])

Creates new NTemplate (see page 695) object.

C#

```
public NTemplate AddPalms(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	The packed NTemplate (see page 695) object.

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.7.3 NTemplate.AddPalms Method (byte[], uint)

Creates new NTemplate (see page 695) object.

C#

```
public NTemplate AddPalms(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	The packed NTemplate (see page 695) object.
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. Must be zero.

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.7.4 NTemplate.AddPalms Method (uint)

Creates new NTemplate (see page 695) object.

C#

```
public NTemplate AddPalms(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method. Must be zero.

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.8 NTemplate.AddPalmsCopy Method

Creates one NTemplate (see page 695) object from another.

C#

```
public NTemplate AddPalmsCopy(NTemplate srcPalms);
```

Parameters

Parameters	Description
NTemplate srcPalms	The NTemplate (see page 695) object.

Returns

The NTemplate (see page 695) object.

6.2.2.1.17.2.9 NTemplate.CalculateSize Method

Calculates the size of a packed NTemplate (see page 726) containing fingers, faces, irises and palms templates of the specified size.

C#

```
public static int CalculateSize(int fingersTemplateSize, int facesTemplateSize, int irisesTemplateSize, int palmsTemplateSize);
```

Parameters

Parameters	Description
int fingersTemplateSize	Size of packed NTemplate (see page 695).

int facesTemplateSize	Size of packed NLTemplate.
int irisesTemplateSize	Size of packed NETemplate.
int palmsTemplateSize	Size of packed NFTemplate (see page 695).

Returns

The value of NTemplate (see page 726) size.

6.2.2.1.17.2.10 Check Method**6.2.2.1.17.2.10.1 NTemplate.Check Method (IntPtr, int)**

Checks if format of packed NTemplate (see page 726) format is correct.

C#

```
public static void Check(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NTemplate (see page 726).
int bufferSize	Size of memory buffer that contains packed NTemplate (see page 726).

Returns

true if NTemplate (see page 726) format is correct; false otherwise.

6.2.2.1.17.2.10.2 NTemplate.Check Method (byte[])

Checks if format of packed NTemplate (see page 726) format is correct.

C#

```
public static void Check(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Memory buffer that contains packed NTemplate (see page 726).

Returns

True if NTemplate (see page 726) format is correct; false otherwise.

6.2.2.1.17.2.11 NTemplate.Clear Method

Removes all NFRecord (see page 666) objects.

C#

```
public void Clear();
```

6.2.2.1.17.2.12 NTemplate.Clone Method

Creates NTemplate (see page 726) object from another NTemplate (see page 726) object.

C#

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

Returns

A NTemplate (see page 726) object.

6.2.2.1.17.2.13 FromHandle Method

6.2.2.1.17.2.13.1 NTemplate.FromHandle Method (IntPtr)

Creates NTemplate (see page 726) object from handle.

C#

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

Parameters

Parameters	Description
IntPtr handle	Handle (see page 623) to unmanaged NTemplate (see page 726) object.

Returns

A NTemplate (see page 726) object.

6.2.2.1.17.2.13.2 NTemplate.FromHandle Method (IntPtr, bool)

Creates NTemplate (see page 726) object from handle.

C#

```
public static NTemplate FromHandle(IntPtr handle, bool ownsHandle);
```

Parameters

Parameters	Description
IntPtr handle	Handle (see page 623) to unmanaged NTemplate (see page 726) object.
bool ownsHandle	If set to true, on destruction of the NTemplate (see page 726) the NTemplate (see page 726) pointed to by the handle will also be destroyed.

Returns

A NTemplate (see page 726) object.

6.2.2.1.17.2.14 GetSize Method

6.2.2.1.17.2.14.1 NTemplate.GetSize Method ()

Calculates packed size of NTemplate (see page 695) object.

C#

```
public int GetSize();
```

Returns

Size of NTemplate (see page 726) object.

6.2.2.1.17.2.14.2 NTemplate.GetSize Method (uint)

Calculates packed size of NTemplate (see page 695) object. Behavior is controlled through flags.

C#

```
public int GetSize(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that controls behavior of the method.

Returns

Size of NTemplate (see page 726) object.

Remarks

The flag FlagProcessFirstRecordOnly is supported.

6.2.2.1.17.2.15 Pack Method**6.2.2.1.17.2.15.1 NTemplate.Pack Method (IntPtr, int, IntPtr, int, IntPtr, int, IntPtr, int)**

Packs packed fingers template as NTemplate (see page 726) into the specified memory buffer.

C#

```
public static byte[] Pack(IntPtr pFingersTemplate, int fingersTemplateSize, IntPtr
pFacesTemplate, int facesTemplateSize, IntPtr pIrisesTemplate, int irisesTemplateSize,
IntPtr pPalmsTemplate, int palmsTemplateSize);
```

Parameters

Parameters	Description
IntPtr pFingersTemplate	Memory buffer with packed NTemplate (see page 695).
int fingersTemplateSize	NTemplate (see page 695) size of memory buffer.
IntPtr pFacesTemplate	Memory buffer with packed NTemplate.
int facesTemplateSize	NTemplate size of memory buffer.
IntPtr pIrisesTemplate	Memory buffer with packed NTemplate.
int irisesTemplateSize	NTemplate size of memory buffer.
IntPtr pPalmsTemplate	Memory buffer with packed NTemplate (see page 695).
int palmsTemplateSize	NTemplate (see page 695) size of memory buffer.

Returns

A byte array with packed NTemplate (see page 726) objects.

6.2.2.1.17.2.15.2 NTemplate.Pack Method (byte[], byte[], byte[], byte[])

Packs packed fingers template as NTemplate (see page 726) into the specified memory buffer.

C#

```
public static byte[] Pack(byte[] fingersTemplate, byte[] facesTemplate, byte[]
irisesTemplate, byte[] palmsTemplate);
```

Parameters

Parameters	Description
byte[] fingersTemplate	The byte array with packed NTemplate (see page 695).
byte[] facesTemplate	The byte array with packed NTemplate.
byte[] irisesTemplate	The byte array with packed NTemplate.
byte[] palmsTemplate	The byte array with packed NTemplate (see page 695).

Returns

A byte array with packed NTemplate (see page 726) objects.

6.2.2.1.17.2.16 NTemplate.RemoveFaces Method

Removes faces template from the NTemplate (see page 726).

C#

```
public void RemoveFaces();
```

6.2.2.1.17.2.17 NTemplate.RemoveFingers Method

Removes fingers template from the NTemplate (see page 726).

C#

```
public void RemoveFingers();
```

6.2.2.1.17.2.18 NTemplate.RemoveIrises Method

Removes irises template from the NTemplate (see page 726).

C#

```
public void RemoveIrises();
```

6.2.2.1.17.2.19 NTemplate.RemovePalms Method

Removes palms template from the NTemplate (see page 726).

C#

```
public void RemovePalms();
```

6.2.2.1.17.2.20 Save Method**6.2.2.1.17.2.20.1 NTemplate.Save Method ()**

Packs NTemplate (see page 726) object to byte array.

C#

```
public byte[] Save();
```

Returns

A byte array with packed NTemplate (see page 726) object.

6.2.2.1.17.2.20.2 NTemplate.Save Method (IntPtr, int)

Packs NTemplate (see page 726) object to memory buffer.

C#

```
public int Save(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer that contains packed NTemplate (see page 726).
int bufferSize	Size of memory buffer.

Returns

The size of packed NTemplate (see page 726)

6.2.2.1.17.2.20.3 NTemplate.Save Method (IntPtr, int, uint)

Packs NTemplate (see page 726) object to memory buffer.

C#

```
public int Save(IntPtr pBuffer, int bufferSize, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer that contains packed NTemplate (see page 726).
int bufferSize	Size of memory buffer.
uint flags	Bitwise combination of zero or more flags that control behavior of the method.

Returns

The size of packed NTemplate (see page 726)

Remarks

The following flags are supported: Flags supported by NFRecord (see page 666) Save methods and flags supported by NTemplate (see page 695) Save methods are applied to fingers template (if) contained in the NTemplate (see page 726). Flags supported by NLTemplates Save methods are applied to faces template (if) contained in the NTemplate (see page 726). Flags supported by NETemplates Save methods are applied to irises template (if) contained in the NTemplate (see page 726).

6.2.2.1.17.2.20.4 NTemplate.Save Method (byte[])

Packs NTemplate (see page 726) object to byte array.

C#

```
public int Save(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NTemplate (see page 726).

Returns

The size of packed NTemplate (see page 726)

6.2.2.1.17.2.20.5 NTemplate.Save Method (byte[], uint)

Packs NTemplate (see page 726) object to byte array. Behavior is controlled through flags.

C#

```
public int Save(byte[] buffer, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains packed NTemplate (see page 726).
uint flags	Bitwise combination of zero or more flags that control behavior of the method.

Returns

The size of packed NTemplate (see page 726)

Remarks

The following flags are supported: Flags supported by NFRecord (see page 666) Save methods and flags supported by NTemplate (see page 695) Save methods are applied to fingers template (if) contained in the NTemplate (see page 726).

726). Flags supported by NLTemplates Save methods are applied to faces template (if) contained in the NTemplate (see page 726). Flags supported by NETemplates Save methods are applied to irises template (if) contained in the NTemplate (see page 726).

6.2.2.1.17.2.20.6 NTemplate.Save Method (uint)

Packs NTemplate (see page 726) object to byte array. Behavior is controlled through flags.

C#

```
public byte[] Save(uint flags);
```

Parameters

Parameters	Description
uint flags	Bitwise combination of zero or more flags that control behavior of the method.

Returns

A byte array with packed NTemplate (see page 726) object.

Remarks

The following flags are supported: Flags supported by NFRecord (see page 666) Save methods and flags supported by NFTemplate (see page 695) Save methods are applied to fingers template (if) contained in the NTemplate (see page 726). Flags supported by NLTemplates Save methods are applied to faces template (if) contained in the NTemplate (see page 726). Flags supported by NETemplates Save methods are applied to irises template (if) contained in the NTemplate (see page 726).

6.2.2.1.17.2.21 Unpack Method

6.2.2.1.17.2.21.1 NTemplate.Unpack Method (IntPtr, int, out byte, out byte, out uint, out byte, out IntPtr, out int, out IntPtr, out int, out IntPtr, out int, out IntPtr, out int)

Unpacks packed fingers template from the packed NTemplate (see page 726).

C#

```
public static void Unpack(IntPtr pBuffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr pFingersTemplate, out int fingersTemplateSize, out IntPtr pFacesTemplate, out int facesTemplateSize, out IntPtr pIrisesTemplate, out int irisesTemplateSize, out IntPtr pPalmsTemplate, out int palmsTemplateSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NTemplate (see page 726) object.
int bufferSize	Size of memory buffer that contains packed NTemplate (see page 726).
out byte majorVersion	Major version of NTemplate (see page 726).
out byte minorVersion	Minor version of NTemplate (see page 726).
out uint size	The size of buffer from header of packed NTemplate (see page 726). This size is equal to bufferSize.
out byte headerSize	The size of packed NTemplate (see page 726) header size.
out IntPtr pFingersTemplate	Pointer to array that contains packed NFTemplate (see page 695).
out int fingersTemplateSize	The size of packed NFTemplate (see page 695).

out IntPtr pFacesTemplate	Pointer to array that contains packed NLTemplate.
out int facesTemplateSize	The size of packed NLTemplate.
out IntPtr pIrisTemplate	Pointer to array that contains packed NETemplate.
out int irisesTemplateSize	The size of packed NETemplate.
out IntPtr pPalmsTemplate	Pointer to array that contains packed NFTemplate (see page 695).
out int palmsTemplateSize	The size of packed NFTemplate (see page 695).

6.2.2.1.17.2.21.2 NTemplate.Unpack Method (IntPtr, int, out byte, out byte, out uint, out byte, out byte[], out byte[], out byte[], out byte[])

Unpacks packed fingers template from the packed NTemplate (see page 726).

C#

```
public static void Unpack(IntPtr pBuffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[] fingersTemplate, out byte[] facesTemplate, out byte[] irisesTemplate, out byte[] palmsTemplate);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains packed NTemplate (see page 726) object.
int bufferSize	Size of memory buffer that contains packed NTemplate (see page 726).
out byte majorVersion	Major version of NTemplate (see page 726).
out byte minorVersion	Minor version of NTemplate (see page 726).
out uint size	The size of buffer from header of packed NTemplate (see page 726). This size is equal to bufferSize.
out byte headerSize	The size of packed NTemplate (see page 726) header size.
out byte[] fingersTemplate	Byte array that contains packed NTemplate (see page 695).
out byte[] facesTemplate	Byte array with packed NLTemplate.
out byte[] irisesTemplate	Byte array with packed NETemplate.
out byte[] palmsTemplate	Byte array with packed NTemplate (see page 695).

6.2.2.1.17.2.21.3 NTemplate.Unpack Method (byte[], out byte, out byte, out uint, out byte, out IntPtr, out int, out IntPtr, out int, out IntPtr, out int)

Unpacks packed fingers template from the packed NTemplate (see page 726).

C#

```
public static void Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr pFingersTemplate, out int fingersTemplateSize, out IntPtr pFacesTemplate, out int facesTemplateSize, out IntPtr pIrisTemplate, out int irisesTemplateSize, out IntPtr pPalmsTemplate, out int palmsTemplateSize);
```

Parameters

Parameters	Description
byte[] buffer	A byte array that contains packed NTemplate (see page 726).
out byte majorVersion	Major version of NTemplate (see page 726).
out byte minorVersion	Minor version of NTemplate (see page 726).

out uint size	The size of buffer from header of packed NTemplate (see page 726).
out byte headerSize	The size of packed NTemplate (see page 726) header size.
out IntPtr pFingersTemplate	Pointer to array that contains packed NTemplate (see page 695).
out int fingersTemplateSize	The size of packed NTemplate (see page 695).
out IntPtr pFacesTemplate	Pointer to array that contains packed NLTemplate.
out int facesTemplateSize	The size of packed NLTemplate.
out IntPtr pIrisTemplate	Pointer to array that contains packed NETemplate.
out int irisTemplateSize	The size of packed NETemplate.
out IntPtr pPalmsTemplate	Pointer to array that contains packed NTemplate (see page 695).
out int palmsTemplateSize	The size of packed NTemplate (see page 695).

6.2.2.1.17.2.21.4 NTemplate.Unpack Method (byte[], out byte, out byte, out uint, out byte, out byte[], out byte[], out byte[], out byte[])

Unpacks packed fingers template from the packed NTemplate (see page 726).

C#

```
public static void Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out
uint size, out byte headerSize, out byte[] fingersTemplate, out byte[] facesTemplate, out
byte[] irisTemplate, out byte[] palmsTemplate);
```

Parameters

Parameters	Description
byte[] buffer	A byte array that contains packed NTemplate (see page 726).
out byte majorVersion	Major version of NTemplate (see page 726).
out byte minorVersion	Minor version of NTemplate (see page 726).
out uint size	The size of buffer from header of packed NTemplate (see page 726).
out byte headerSize	The size of packed NTemplate (see page 726) header size.
out byte[] fingersTemplate	Byte array that contains packed NTemplate (see page 695).
out byte[] facesTemplate	Byte array with packed NLTemplate.
out byte[] irisTemplate	Byte array with packed NETemplate.
out byte[] palmsTemplate	Byte array with packed NTemplate (see page 695).

6.2.2.1.17.3 NTemplate Properties

6.2.2.1.17.3.1 NTemplate.Faces Property

Gets NLTemplate object.

C#

```
public NLTemplate Faces;
```

Property value

A NLTemplate object.

6.2.2.1.17.3.2 NTemplate.Fingers Property

Gets NTemplate (see page 695) object.

C#

```
public NTemplate Fingers;
```

Property value

A NTemplate ([see page 695](#)) object.

6.2.2.1.17.3.3 NTemplate.Irises Property

Gets NTemplate object.

C#

```
public NTemplate Irises;
```

Property value

A NTemplate object.

6.2.2.1.17.3.4 NTemplate.Palms Property

Gets NTemplate ([see page 695](#)) object.

C#

```
public NTemplate Palms;
```

Property value

A NTemplate ([see page 695](#)) object.

6.2.2.1.18 NTemplateInfo Class

For internal use.

Class Hierarchy**C#**

```
public class NTemplateInfo : NInfo;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NTemplateInfo Properties**NTemplateInfo Class**

	Name	Description
	FacesInfo (see page 745)	For internal use only.
	FingersInfo (see page 745)	For internal use only.
	HeaderSize (see page 745)	For internal use only.
	IrisesInfo (see page 745)	For internal use only.
	MajorVersion (see page 745)	For internal use only.
	MinorVersion (see page 745)	For internal use only.
	PalmsInfo (see page 745)	For internal use only.
	Size (see page 745)	For internal use only.

6.2.2.1.18.1 NTemplateInfo Properties

6.2.2.1.18.1.1 NTemplateInfo.FacesInfo Property

For internal use only.

C#

```
public NTemplateInfo FacesInfo;
```

6.2.2.1.18.1.2 NTemplateInfo.FingersInfo Property

For internal use only.

C#

```
public NTemplateInfo FingersInfo;
```

6.2.2.1.18.1.3 NTemplateInfo.HeaderSize Property

For internal use only.

C#

```
public byte HeaderSize;
```

6.2.2.1.18.1.4 NTemplateInfo.IrisesInfo Property

For internal use only.

C#

```
public NTemplateInfo IrisesInfo;
```

6.2.2.1.18.1.5 NTemplateInfo.MajorVersion Property

For internal use only.

C#

```
public byte MajorVersion;
```

6.2.2.1.18.1.6 NTemplateInfo.MinorVersion Property

For internal use only.

C#

```
public byte MinorVersion;
```

6.2.2.1.18.1.7 NTemplateInfo.PalmsInfo Property

For internal use only.

C#

```
public NTemplateInfo PalmsInfo;
```

6.2.2.1.18.1.8 NTemplateInfo.Size Property

For internal use only.

C#

```
public uint Size;
```

6.2.2.1.19 NTemplates Class

Provides information about the NTemplates (see page 605) library.

Class Hierarchy

C#

```
public static class NTemplates;
```

NTemplates Fields

	Name	Description
	DllName ( see page 746)	Name of DLL containing unmanaged part of this class.

NTemplates Methods

	Name	Description
	GetInfo ( see page 746)	Retrieves information about the library.

6.2.2.1.19.1 NTemplates Fields

6.2.2.1.19.1.1 NTemplates.DllName Field

Name of DLL containing unmanaged part of this class.

C#

```
public const string DllName = "NTemplates";
```

6.2.2.1.19.2 NTemplates Methods

6.2.2.1.19.2.1 NTemplates.GetInfo Method

Retrieves information about the library.

C#

```
public static NLibraryInfo GetInfo();
```

Returns

Returns NLibraryInfo ( see page 619) structure with library information.

6.2.2.2 Structs, Records, Enums

6.2.2.2.1 Neurotec.Biometrics.NBiometricType Enumeration

Specifies the type of biometric data.

C#

```
[Flags]
public enum NBiometricType {
}
```

6.2.2.2.2 Neurotec.Biometrics.NfeExtractionStatus Enumeration

Specifies fingerprint features extraction status.

C#

```
[Serializable]
public enum NfeExtractionStatus {
    TemplateCreated = 1,
    TooFewMinutiae = 90,
    QualityCheckFailed = 100,
    MatchingFailed = 200
}
```

Members

Members	Description
TemplateCreated = 1	Features extracted successfully.
TooFewMinutiae = 90	Extracted minutiae count is less than MinMinutiaCount.
QualityCheckFailed = 100	Fingerprint image is low quality.
MatchingFailed = 200	During generalization score between two templates was less than required threshold.

6.2.2.2.3 Neurotec.Biometrics.NfeReturnedImage Enumeration

Specifies the returned image.

C#

```
[Serializable]
public enum NfeReturnedImage {
    None = 0,
    Binarized = 1,
    Skeletonized = 2
}
```

Members

Members	Description
None = 0	No image.
Binarized = 1	The binarized image.
Skeletonized = 2	The skeletonized image.

6.2.2.2.4 Neurotec.Biometrics.NfeTemplateSize Enumeration

Specifies a returned NFRecord ([see page 666](#)) size from methods: Extract, ExtractUnpacked.

C#

```
[Serializable]
public enum NfeTemplateSize {
    Small = 0,
    Large = 256
}
```

Members

Members	Description
Small = 0	The small NFRecord (see page 666) size.
Large = 256	The large NFRecord (see page 666) size.

6.2.2.2.5 Neurotec.Biometrics.NFImpressionType Enumeration

Specifies the impression types.

C#

```
public enum NFImpressionType {
    LiveScanPlain = 0,
    LiveScanRolled = 1,
    NonLiveScanPlain = 2,
    NonLiveScanRolled = 3,
    LatentImpression = 4,
    LatentTracing = 5,
    LatentPhoto = 6,
    LatentLift = 7,
    Swipe = 8,
}
```

```

    LiveScanContactless = 9,
    LiveScanPalm = 10,
    NonLiveScanPalm = 11,
    LatentPalmImpression = 12,
    LatentPalmTracing = 13,
    LatentPalmPhoto = 14,
    LatentPalmLift = 15
}

```

Members

Members	Description
LiveScanPlain = 0	Live-scanned plain fingerprint.
LiveScanRolled = 1	Live-scanned rolled fingerprint.
NonliveScanPlain = 2	Nonlive-scanned (from paper) plain fingerprint.
NonliveScanRolled = 3	Nonlive-scanned (from paper) rolled fingerprint.
LatentImpression = 4	Latent impression fingerprint.
LatentTracing = 5	Latent tracing fingerprint.
LatentPhoto = 6	Latent photo fingerprint.
LatentLift = 7	Latent lift fingerprint.
Swipe = 8	Live-scanned fingerprint by sliding the finger across a "swipe" sensor.
LiveScanContactless = 9	Live-scanned fingerprint using contactless device.
LiveScanPalm = 10	Live scanned palm.
NonliveScanPalm = 11	Non-live scanned palm.
LatentPalmImpression = 12	Latent palm impression.
LatentPalmTracing = 13	Latent palm tracing.
LatentPalmPhoto = 14	Latent palm photo.
LatentPalmLift = 15	Latent palm lift.

6.2.2.2.6 Neurotec.Biometrics.NFMinutiaFormat Enumeration

Specifies formats of minutia. This enumeration allows a bitwise combination of its member values.

C#

```

[Flags]
public enum NFMinutiaFormat {
    None = 0,
    HasQuality = 1,
    HasCurvature = 2,
    HasG = 4
}

```

Members

Members	Description
None = 0	None of other values.
HasQuality = 1	Indicates that Neurotec.Biometrics.NFMinutia.Quality (see page 663) field contains meaningful value and is preserved during unpacking/packing of NFRecord (see page 666).
HasCurvature = 2	Indicates that Neurotec.Biometrics.NFMinutia.Curvature (see page 663) field contains meaningful value and is preserved during unpacking/packing of NFRecord (see page 666).
HasG = 4	Indicates that Neurotec.Biometrics.NFMinutia.G (see page 663) field contains meaningful value and is preserved during unpacking/packing of NFRecord (see page 666).

6.2.2.2.7 Neurotec.Biometrics.NFMinutiaOrder Enumeration

Specifies minutia order.

C#

```
[Flags]
public enum NFMinutiaOrder {
    Ascending = 0x01,
    Descending = 0x02,
    CartesianXY = 0x04,
    CartesianYX = 0x08,
    Angle = 0x0C,
    Polar = 0x10
}
```

Members

Members	Description
Ascending = 0x01	Specifies than minutiae are sorted ascending by the specified order.
Descending = 0x02	Specifies than minutiae are sorted descending by the specified order.
CartesianXY = 0x04	Specifies than minutiae are sorted by X field. If X field of two minutiae are equal Y field is compared.
CartesianYX = 0x08	Specifies than minutiae are sorted by Y field. If Y field of two minutiae are equal X field is compared.
Angle = 0x0C	Specifies than minutiae are sorted by Angle field.
Polar = 0x10	Specifies than minutiae are sorted by distance from minutiae center of mass. If distance of two minutiae are equal Angle field is compared.

6.2.2.2.8 Neurotec.Biometrics.NFMinutiaType Enumeration

Specifies types of minutia.

C#

```
public enum NFMinutiaType {
    Unknown = 0,
    End = 1,
    Bifurcation = 2,
    Other = 3
}
```

Members

Members	Description
Unknown = 0	The type of the minutia is unknown.
End = 1	The minutia that is an end of a ridge.
Bifurcation = 2	The minutia that is a bifurcation of a ridge.
Other = 3	Other.

6.2.2.2.9 Neurotec.Biometrics.NFPatternClass Enumeration

Specifies pattern class of the fingerprint.

C#

```
public enum NFPatternClass {
    Unknown = 0,
    PlainArch = 1,

```

```

    TentedArch = 2,
    RadialLoop = 3,
    UlnarLoop = 4,
    PlainWhorl = 5,
    CentralPocketLoop = 6,
    DoubleLoop = 7,
    AccidentalWhorl = 8,
    Whorl = 9,
    RightSlantLoop = 10,
    LeftSlantLoop = 11,
    Scar = 12,
    Amputation = 15
}

```

Members

Members	Description
Unknown = 0	Unknown pattern class.
PlainArch = 1	Plain arch pattern class.
TentedArch = 2	Tented arch pattern class.
RadialLoop = 3	Radial loop pattern class.
UlnarLoop = 4	Ulnar loop pattern class.
PlainWhorl = 5	Plain whorl pattern class.
CentralPocketLoop = 6	Central pocket loop pattern class.
DoubleLoop = 7	Double loop pattern class.
AccidentalWhorl = 8	Accidental whorl pattern class.
Whorl = 9	Whorl pattern class.
RightSlantLoop = 10	Right slant loop pattern class.
LeftSlantLoop = 11	Left slant loop pattern class.
Scar = 12	Scar. Pattern class is not available.
Amputation = 15	Amputation. Pattern class is not available.

Remarks

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

6.2.2.2.10 Neurotec.Biometrics.NFPosition Enumeration

Specifies finger position.

C#

```

public enum NFPosition {
    Unknown = 0,
    RightThumb = 1,
    RightIndex = 2,
    RightMiddle = 3,
    RightRing = 4,
    RightLittle = 5,
    LeftThumb = 6,
    LeftIndex = 7,
    LeftMiddle = 8,
    LeftRing = 9,
    LeftLittle = 10,
    PlainRightThumb = 11,
    PlainLeftThumb = 12,
    PlainRightFourFingers = 13,
    PlainLeftFourFingers = 14,
    PlainThumbs = 15,
    UnknownPalm = 20,
    RightFullPalm = 21,
    RightWritersPalm = 22,
    LeftFullPalm = 23,
}

```

```

    LeftWritersPalm = 24,
    RightLowerPalm = 25,
    RightUpperPalm = 26,
    LeftLowerPalm = 27,
    LeftUpperPalm = 28,
    RightOther = 29,
    LeftOther = 30,
    RightInterdigital = 31,
    RightThenar = 32,
    RightHypothenar = 33,
    LeftInterdigital = 34,
    LeftThenar = 35,
    LeftHypothenar = 36
}

```

Members

Members	Description
Unknown = 0	Unknown finger.
RightThumb = 1	Thumb of the right hand.
RightIndex = 2	Index finger of the right hand.
RightMiddle = 3	Middle finger of the right hand.
RightRing = 4	Ring finger of the right hand.
RightLittle = 5	Little finger of the right hand.
LeftThumb = 6	Thumb of the left hand.
LeftIndex = 7	Index finger of the left hand.
LeftMiddle = 8	Middle finger of the left hand.
LeftRing = 9	Ring finger of the left hand.
LeftLittle = 10	Little finger of the left hand.
PlainRightThumb = 11	Specifies plain right thumb.
PlainLeftThumb = 12	Specifies leftplain thumb.
PlainRightFourFingers = 13	Specifies plain right four fingers.
PlainLeftFourFingers = 14	Specifies left four fingers.
PlainThumbs = 15	Specifies plain thumbs.
UnknownPalm = 20	Unknown palm.
RightFullPalm = 21	Specifies full left palm.
RightWritersPalm = 22	Specifies right writers palm.
LeftFullPalm = 23	Specifies full left palm.
LeftWritersPalm = 24	Specifies left writers palm.
RightLowerPalm = 25	Specifies right lower palm.
RightUpperPalm = 26	Specifies right upper palm.
LeftLowerPalm = 27	Specifies lower palm.
LeftUpperPalm = 28	Specifies right upper palm.
RightOther = 29	Some other right palm.
LeftOther = 30	Some other left hand palm.
RightInterdigital = 31	Specifies right interdigital.
RightThenar = 32	Specifies right thenar.
RightHypothenar = 33	Specifies right hypothenar.
LeftInterdigital = 34	Specifies left interdigital.
LeftThenar = 35	Specifies left thenar.
LeftHypothenar = 36	Specifies left hypothenar.

6.2.2.2.11 Neurotec.Biometrics.NFRidgeCountsType Enumeration

Specifies type of ridge counts contained in NfRecord (see page 666).

C#

```
public enum NFRidgeCountsType {
    None = 0,
    FourNeighbors = 1,
    EightNeighbors = 2,
    FourNeighborsWithIndexes = 5,
    EightNeighborsWithIndexes = 6,
    Unspecified = 128+4
}
```

Members

Members	Description
None = 0	The NfRecord (see page 666) does not contain ridge counts.
FourNeighbors = 1	The NfRecord (see page 666) contains ridge counts to closest minutia in each of the four sectors of each minutia. First sector starts at minutia angle.
EightNeighbors = 2	The NfRecord (see page 666) contains ridge counts to closest minutia in each of the eight sectors of each minutia. First sector starts at minutia angle.
FourNeighborsWithIndexes = 5	The NfRecord (see page 666) contains ridge counts to four neighbors of each minutia.
EightNeighborsWithIndexes = 6	The NfRecord (see page 666) contains ridge counts to eight neighbors of each minutia.
Unspecified = 128+4	For internal use.

Remarks

Extracted template with EightNeighborsWithIndexes parameter is bigger than the template extracted with EightNeighbors parameter. Templates extracted with EightNeighborsWithIndexes parameter is faster than the templates extracted with EightNeighbors parameter. Extracted template with FourNeighborsWithIndexes parameter is bigger than the template extracted with FourNeighbors parameter. Templates extracted with FourNeighborsWithIndexes parameter is faster than the templates extracted with FourNeighbors parameter.

6.2.2.2.12 Neurotec.Biometrics.NleExtractionStatus Enumeration

Specifies face features extraction status.

C#

```
[Serializable]
public enum NleExtractionStatus {
    None = 0,
    TemplateCreated = 1,
    FaceNotDetected = 2,
    EyesNotDetected = 3,
    FaceTooCloseToImageBorder = 4,
    QualityCheckGrayscaleDensityFailed = 100,
    QualityCheckExposureFailed = 101,
    QualityCheckSharpnessFailed = 102,
    LivenessCheckFailed = 200,
    GeneralizationFailed = 300
}
```

Members

Members	Description
None = 0	No extraction status usage.
TemplateCreated = 1	Template created.
FaceNotDetected = 2	Face not detected.
EyesNotDetected = 3	Eyes not detected.
FaceTooCloseToImageBorder = 4	Face too close to image border.
QualityCheckGrayscaleDensityFailed = 100	Quality check grayscale density failed.
QualityCheckExposureFailed = 101	Quality check exposure failed.
QualityCheckSharpnessFailed = 102	Quality check sharpness failed.
LivenessCheckFailed = 200	Liveness check failed.
GeneralizationFailed = 300	Generalization failed.

6.2.2.2.13 Neurotec.Biometrics.NleTemplateSize Enumeration

Specifies NLRecord type by its size.

C#

```
[Serializable]
public enum NleTemplateSize {
    Small = 0,
    Medium = 64,
    Large = 128
}
```

Members

Members	Description
Small = 0	The small NLRecord size (~2.2KB).
Medium = 64	The small-medium NLRecord size (~11KB).
Large = 128	The medium NLRecord size (~20KB).

6.2.3 Neurotec.Biometrics.Standards Namespace

This namespace contains classes of supported biometrical standards.

Classes

	Name	Description
	Bdif (see page 754)	Provides library registration and other additional functionality.
	BdifFaceFeaturePoint (see page 755)	Provides ...
	BdifFPMinutiaNeighbor (see page 756)	Provides ...
	CbeffBdbFormatIdentifiers (see page 758)	Specifies CBEFF Biometric Data Block (BDB) Format identifiers registered with International Biometric Industry Association (IBIA).
	CbeffBiometricOrganizations (see page 776)	Specifies CBEFF biometric organizations identifiers registered with International Biometric Industry Association (IBIA).
	FMCard (see page 788)	Provides functionality for converting ISO/IEC 19794-2:2005 card format standard templates to and from Neurotechnology Finger Records (NFRecords).
	FmcBdtTags (see page 801)	Provides constants for ISO/IEC 7816-11:2004 Biometric data template.

	FmcBitBhtBaptTags (see page 802)	Provides constants for ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
	NBiometricStandards (see page 803)	Provides methods for retrieving library information.

Module

.NET Reference ([see page 606](#))

Structs, Records, Enums

	Name	Description
	BdifEyeColor (see page 804)	Enumerates eye color constants.
	BdifEyePosition (see page 804)	Enumerates eye position constants.
	BdifFaceExpression (see page 805)	Enumerates facial feature constants.
	BdifFaceFeaturePointType (see page 805)	Enumerates face feature point type constants.
	BdifFaceProperties (see page 805)	Enumerates basic face properties constants
	BdifFPatternClass (see page 806)	Enumerates fingerprint pattern class constants.
	BdifFPImpressionType (see page 807)	Enumerates fingerprint impression type constants.
	BdifFPMinutiaType (see page 808)	Enumerates fingerprint minutia type constants.
	BdifFPPosition (see page 809)	Enumerates fingerprint position constants.
	BdifGender (see page 810)	Enumerates gender constants.
	BdifHairColor (see page 811)	Enumerates hair color constants.
	BdifImageSourceType (see page 811)	Enumerates image source type constants
	BdifIrisOrientation (see page 812)	Enumerates Bdif (see page 754) Iris orientation values.
	BdifIrisScanType (see page 812)	Enumerates Bdif (see page 754) scan type values.
	BdifScaleUnits (see page 813)	Enumerates scale unit constants.
	BdifStandard (see page 813)	Enumerates standard constants.
	FmcFeatureHandling (see page 813)	Specifies additional fingerprint features that are supported by the card.
	FmcMinutiaOrder (see page 814)	Specifies minutia ordering required by the card.

6.2.3.1 Classes

6.2.3.1.1 Bdif Class

Provides library registration and other additional functionality.

Class Hierarchy

C#

```
public static class Bdif;
```

Remarks

Neurotec.Biometrics.Bdif.dll library provides functionality of this library. Neurotec.Biometrics.Bdif.dll is a wrapper of the Bdif.dll.

Bdif Fields

	Name	Description
	FlagDoNotCheckCbeffProductId (see page 755)	This flag is used when Cbeff product Id should not be checked.

6.2.3.1.1.1 Bdif Fields**6.2.3.1.1.1.1 Bdif.FlagDoNotCheckCbeffProductId Field**

This flag is used when Cbeff product Id should not be checked.

C#

```
public const uint FlagDoNotCheckCbeffProductId = 0x00000002;
```

6.2.3.1.2 BdifFaceFeaturePoint Structure

Provides ...

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct BdifFaceFeaturePoint {
}
```

Methods

	Name	Description
	BdifFaceFeaturePoint (see page 755)	Initializes a new instance of the BdifFaceFeaturePoint class.

BdifFaceFeaturePoint Methods

	Name	Description
	ToString (see page 756)	Returns a string that represents the current structure.

BdifFaceFeaturePoint Properties

	Name	Description
	Code (see page 756)	Code.
	Y (see page 756)	Gets or sets Y coordinate of the face feature point.
	Type (see page 756)	Gets or sets the type of the face feature point.
	X (see page 756)	Gets or sets X coordinate of the face feature point.

6.2.3.1.2.1 BdifFaceFeaturePoint.BdifFaceFeaturePoint Constructor

Initializes a new instance of the BdifFaceFeaturePoint class.

C#

```
public BdifFaceFeaturePoint(byte code, ushort x, ushort y);
```

Parameters

Parameters	Description
byte code	Code (see page 756).
ushort x	X (see page 756) coordinate of the face feature point
ushort y	Y (see page 756) coordinate of the face feature point.

6.2.3.1.2.2 BdifFaceFeaturePoint Methods

6.2.3.1.2.2.1 BdifFaceFeaturePoint.ToString Method

Returns a string that represents the current structure.

C#

```
public override string ToString();
```

Returns

String representation of the BdifFaceFeaturePoint (see page 755) object.

6.2.3.1.2.3 BdifFaceFeaturePoint Properties

6.2.3.1.2.3.1 BdifFaceFeaturePoint.Code Property

Code.

C#

```
public byte Code;
```

Property value

Code.

6.2.3.1.2.3.2 BdifFaceFeaturePoint.Y Property

Gets or sets Y coordinate of the face feature point.

C#

```
public ushort Y;
```

Property value

Y coordinate.

6.2.3.1.2.3.3 BdifFaceFeaturePoint.Type Property

Gets or sets the type of the face feature point.

C#

```
public BdifFaceFeaturePointType Type;
```

Property value

Type of face feature point.

6.2.3.1.2.3.4 BdifFaceFeaturePoint.X Property

Gets or sets X coordinate of the face feature point.

C#

```
public ushort X;
```

Property value

X coordinate.

6.2.3.1.3 BdifFPMinutiaNeighbor Structure

Provides ...

C#

```
[StructLayout(LayoutKind.Sequential)]  
public struct BdifFPMinutiaNeighbor {  
    public static readonly BdifFPMinutiaNeighbor Empty = new BdifFPMinutiaNeighbor(-1, 255);  
}
```

BdifFPMinutiaNeighbor Fields

	Name	Description
	Empty (see page 757)	This field is used for an empty BdifFPMinutiaNeighbor object

Methods

	Name	Description
	BdifFPMinutiaNeighbor (see page 757)	Initializes a new instance of the BdifFPMinutiaNeighbor class.

BdifFPMinutiaNeighbor Methods

	Name	Description
	ToString (see page 757)	Returns a string that represents the current structure.

BdifFPMinutiaNeighbor Properties

	Name	Description
	Index (see page 758)	Gets or sets the index of minutia neighbor.
	RidgeCount (see page 758)	Gets or sets the ridge count between the minutia and minutia neighbor.

6.2.3.1.3.1 BdifFPMinutiaNeighbor.BdifFPMinutiaNeighbor Constructor

Initializes a new instance of the BdifFPMinutiaNeighbor class.

C#

```
public BdifFPMinutiaNeighbor(int index, byte ridgeCount);
```

Parameters

Parameters	Description
int index	Minutia index.
byte ridgeCount	Ridge count.

6.2.3.1.3.2 BdifFPMinutiaNeighbor Fields**6.2.3.1.3.2.1 BdifFPMinutiaNeighbor.Empty Field**

This field is used for an empty BdifFPMinutiaNeighbor (see page 756) object

C#

```
public static readonly BdifFPMinutiaNeighbor Empty = new BdifFPMinutiaNeighbor(-1, 255);
```

6.2.3.1.3.3 BdifFPMinutiaNeighbor Methods**6.2.3.1.3.3.1 BdifFPMinutiaNeighbor.ToString Method**

Returns a string that represents the current structure.

C#

```
public override string ToString();
```

Returns

String representation of the BdifFPMinutiaNeighbor (see page 756) object.

6.2.3.1.3.4 BdifFPMinutiaNeighbor Properties

6.2.3.1.3.4.1 BdifFPMinutiaNeighbor.Index Property

Gets or sets the index of minutia neighbor.

C#

```
public int Index;
```

Property value

The index of minutia neighbor.

6.2.3.1.3.4.2 BdifFPMinutiaNeighbor.RidgeCount Property

Gets or sets the ridge count between the minutia and minutia neighbor.

C#

```
public byte RidgeCount;
```

Property value

The ridge count between the minutia and minutia neighbor.

6.2.3.1.4 CbeffBdbFormatIdentifiers Class

Specifies CBEFF Biometric Data Block (BDB) Format identifiers registered with International Biometric Industry Association (IBIA).

Class Hierarchy

C#

```
public static class CbeffBdbFormatIdentifiers;
```

Remarks

To use the functionality of this class you should include Neurotec.Biometrics.Cbeff.dll into your project.

CbeffBdbFormatIdentifiers Fields

	Name	Description
	BioLogicaSistemasIso19794_2BioUniStation (see page 766)	The CBEFF DBD format identifier assigned to BioLogica Sistemas Ltda ISO 19794-2/BioUniStation format.
	BioLogicaSistemasIso19794_4BioUniStation (see page 766)	The CBEFF DBD format identifier assigned to BioLogica Sistemas Ltda ISO 19794-4/BioUniStation format.
	BioLogicaSistemasIso19794_5BioUniStation (see page 766)	The CBEFF DBD format identifier assigned to BioLogica Sistemas Ltda ISO 19794-5/BioUniStation format.
	FederalOfficeForInformationSecurityTRBiometricsXmlFace10 (see page 766)	The CBEFF DBD format identifier assigned to Federal Office for Information Security (BSI) TR Biometrics (see page 639) XML-Face 1.0 format.

	FederalOfficeForInformationSecurityTRBiometricsXmlFinger10 ( see page 766)	The CBEFF DBD format identifier assigned to Federal Office for Information Security (BSI) TR Biometrics ( see page 639) XML-Finger 1.0 format.
	Id3SemiconductorsCompactCardExtension ( see page 767)	The CBEFF DBD format identifier assigned to id3 Semiconductors compact card extension format.
	IncitsTCM1BiometricsAviMedia ( see page 767)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) AVI media file format.
	IncitsTCM1BiometricsBiometricFusionData ( see page 767)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) Biometric Fusion Data Format.
	IncitsTCM1BiometricsBmpImage ( see page 767)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) Microsoft Windows bitmap image format.
	IncitsTCM1BiometricsEbts ( see page 767)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) Electronic Biometric Transmission Specification (EBTS) format.
	IncitsTCM1BiometricsFacelImage ( see page 767)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) Face Image Format.
	IncitsTCM1BiometricsFingerImage ( see page 767)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) Finger Image format.
	IncitsTCM1BiometricsFingerMinutiaeN ( see page 767)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) Finger Minutiae format - no extended data.
	IncitsTCM1BiometricsFingerMinutiaeU ( see page 768)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) Finger Minutiae format - presence of extended data unspecified.

	IncitsTCM1BiometricsFingerMinutiaeX (see page 768)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Finger Minutiae Format - with extended data.
	IncitsTCM1BiometricsFingerPatternN (see page 768)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Finger Pattern format - no extended data.
	IncitsTCM1BiometricsGenericFbiEftsRecord (see page 768)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Generic FBI EFTS record format.
	IncitsTCM1BiometricsGif89aImage (see page 768)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) CompuServe GIF89a image format.
	IncitsTCM1BiometricsHandGeometryN (see page 768)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Hand Geometry format - no extended data.
	IncitsTCM1BiometricsHandGeometryX (see page 768)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Hand Geometry format - with extended data.
	IncitsTCM1BiometricsIrisPolar (see page 768)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Iris Format - polar.
	IncitsTCM1BiometricsIrisRectilinear (see page 769)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Iris Format - rectilinear.
	IncitsTCM1BiometricsJpeg2000Image (see page 769)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) JPEG2000 image format.
	IncitsTCM1BiometricsJpegImage (see page 769)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) JPEG image format.

	IncitsTCM1BiometricsMpeg1Part3Media (see page 769)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) MPEG1 Part 3 media file format.
	IncitsTCM1BiometricsMpegMedia (see page 769)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) MPEG media file format.
	IncitsTCM1BiometricsNiem (see page 769)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) National Information Exchange Model format.
	IncitsTCM1BiometricsNistItl1_2000Type10Record (see page 769)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 10 record format.
	IncitsTCM1BiometricsNistItl1_2000Type13Record (see page 769)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 13 record format.
	IncitsTCM1BiometricsNistItl1_2000Type14Record (see page 770)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 14 record format.
	IncitsTCM1BiometricsNistItl1_2000Type15Record (see page 770)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 15 record format.
	IncitsTCM1BiometricsNistItl1_2000Type16Record (see page 770)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 16 record format.
	IncitsTCM1BiometricsNistItl1_2000Type4Record (see page 770)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 4 record format.
	IncitsTCM1BiometricsNistItl1_2007Type10Record (see page 770)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2007 Type 10 record format.

	IncitsTCM1BiometricsNistItl1_2007Type14Record (see page 770)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2007 Type 14 record format.
	IncitsTCM1BiometricsNistItl1_2007Type17Record (see page 770)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2007 Type 17 record format.
	IncitsTCM1BiometricsNistItl1_200XRecordCollectionForTenPrintCapture (see page 771)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-200x record collection for TenPrint Capture format.
	IncitsTCM1BiometricsPngImage (see page 771)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Portable Network Graphics (PNG) data format.
	IncitsTCM1BiometricsSignatureSignCommonFeatureDataN (see page 771)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - Common feature data - no extended data.
	IncitsTCM1BiometricsSignatureSignCommonFeatureDataX (see page 771)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - Common feature data - with extended data.
	IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataN (see page 771)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - both Raw and Common feature data - no extended data.
	IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataX (see page 771)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - both Raw and Common feature data - with extended data.
	IncitsTCM1BiometricsSignatureSignRawDataN (see page 771)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - Raw data - no extended data.

	IncitsTCM1BiometricsSignatureSignRawDataX ( see page 772)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) Signature/Sign format - Raw data - with extended data.
	IncitsTCM1BiometricsTiffImage ( see page 772)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) TIFF image format.
	IncitsTCM1BiometricsVrml3DObjectData ( see page 772)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) VRML 3D object data format.
	IncitsTCM1BiometricsWavAudio ( see page 772)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) WAV audio file format.
	IncitsTCM1BiometricsWsqImage ( see page 772)	The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639) WSQ compressed fingerprint image format.
	IsolecJtc1SC27ITSecurityTechniquesBiometricReferenceTemplateCertificate ( see page 772)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 27 - IT Security techniques biometric-reference-template-certificate format.
	IsolecJtc1SC37BiometricsFacelImage ( see page 772)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics ( see page 639) face-image format.
	IsolecJtc1SC37BiometricsFingerImage ( see page 772)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics ( see page 639) finger-image format.
	IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactN ( see page 773)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics ( see page 639) finger-minutiae-card-compact-n format.
	IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactRH ( see page 773)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics ( see page 639) finger-minutiae-card-compact-r-h format.

	IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactV (see page 773)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-compact-v format.
	IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactVH (see page 773)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-compact-v-h format.
	IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalN (see page 773)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-normal-n format.
	IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalRNH (see page 773)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics (see page 639) finger-minutiae-card-normal-r-n h format.
	IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalV (see page 773)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-normal-v format.
	IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalVNH (see page 773)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-normal-v-n h format.
	IsolecJtc1SC37BiometricsFingerMinutiaeRecordN (see page 774)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-record-n format.
	IsolecJtc1SC37BiometricsFingerMinutiaeRecordX (see page 774)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-record-x format.
	IsolecJtc1SC37BiometricsFingerPatternSkeletalDataCardCompact (see page 774)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-skeletal-data-card-compact format.

	IsolecJtc1SC37BiometricsFingerPatternSkeletalDataCardNormal (see page 774)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-skeletal-data-card-normal format.
	IsolecJtc1SC37BiometricsFingerPatternSkeletalDataRecord (see page 774)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-skeletal-data-record format.
	IsolecJtc1SC37BiometricsFingerPatternSpectralDiscreteFourierTransform (see page 774)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-spectral-discrete-fourier-transform format.
	IsolecJtc1SC37BiometricsFingerPatternSpectralGaborFilter (see page 774)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-spectral-gabor-filter format.
	IsolecJtc1SC37BiometricsFingerPatternSpectralQuantizedCOSinusoidalTriplet (see page 775)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-spectral-quantized-co-sinusoidal-triplet format.
	IsolecJtc1SC37BiometricsHandGeometrySilhouette (see page 775)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) hand-geometry-silhouette format.
	IsolecJtc1SC37BiometricsIrisImagePolar (see page 775)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) iris-image-polar format.
	IsolecJtc1SC37BiometricsIrisImageRectilinear (see page 775)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) iris-image-rectilinear format.
	IsolecJtc1SC37BiometricsSignatureSignTimeSeriesCompact (see page 775)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) signature-sign-time-series-compact format.

	IsolecJtc1SC37BiometricsSignatureSignTimeSeriesFull (see page 775)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) signature-sign-time-series-full format.
	IsolecJtc1SC37BiometricsVascularImageData (see page 775)	The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) vascular-image-data format.
	NeurotechnologijaNLRecord3 (see page 776)	The CBEFF DBD format identifier assigned by Neurotechnologija for NLRecord version 3 format.
	NeurotechnologijaNTemplate (see page 776)	The CBEFF DBD format identifier assigned by Neurotechnologija for NTemplate (see page 726) format.

6.2.3.1.4.1 CbeffBdbFormatIdentifiers Fields

6.2.3.1.4.1.1 CbeffBdbFormatIdentifiers.BioLogicaSistemasIso19794_2BioUniStation Field

The CBEFF DBD format identifier assigned to BioLogica Sistemas Ltda ISO 19794-2/BioUniStation format.

C#

```
public const ushort BioLogicaSistemasIso19794_2BioUniStation = 0x0002;
```

6.2.3.1.4.1.2 CbeffBdbFormatIdentifiers.BioLogicaSistemasIso19794_4BioUniStation Field

The CBEFF DBD format identifier assigned to BioLogica Sistemas Ltda ISO 19794-4/BioUniStation format.

C#

```
public const ushort BioLogicaSistemasIso19794_4BioUniStation = 0x0004;
```

6.2.3.1.4.1.3 CbeffBdbFormatIdentifiers.BioLogicaSistemasIso19794_5BioUniStation Field

The CBEFF DBD format identifier assigned to BioLogica Sistemas Ltda ISO 19794-5/BioUniStation format.

C#

```
public const ushort BioLogicaSistemasIso19794_5BioUniStation = 0x0005;
```

6.2.3.1.4.1.4 CbeffBdbFormatIdentifiers.FederalOfficeForInformationSecurityTRBiometricsXmlFace10 Field

The CBEFF DBD format identifier assigned to Federal Office for Information Security (BSI) TR Biometrics ([see page 639](#)) XML-Face 1.0 format.

C#

```
public const ushort FederalOfficeForInformationSecurityTRBiometricsXmlFace10 = 0x0001;
```

6.2.3.1.4.1.5 CbeffBdbFormatIdentifiers.FederalOfficeForInformationSecurityTRBiometricsXmlFinger10 Field

The CBEFF DBD format identifier assigned to Federal Office for Information Security (BSI) TR Biometrics ([see page 639](#)) XML-Finger 1.0 format.

C#

```
public const ushort FederalOfficeForInformationSecurityTRBiometricsXmlFinger10 = 0x0002;
```

6.2.3.1.4.1.6 CbeffBdbFormatIdentifiers.Id3SemiconductorsCompactCardExtension Field

The CBEFF DBD format identifier assigned to id3 Semiconductors compact card extension format.

C#

```
public const ushort Id3SemiconductorsCompactCardExtension = 0x0001;
```

6.2.3.1.4.1.7 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsAviMedia Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) AVI media file format.

C#

```
public const ushort IncitsTCM1BiometricsAviMedia = 0x800B;
```

6.2.3.1.4.1.8 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsBiometricFusionData Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Biometric Fusion Data Format.

C#

```
public const ushort IncitsTCM1BiometricsBiometricFusionData = 0x0901;
```

6.2.3.1.4.1.9 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsBmpImage Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Microsoft Windows bitmap image format.

C#

```
public const ushort IncitsTCM1BiometricsBmpImage = 0x8002;
```

6.2.3.1.4.1.10 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsEbts Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Electronic Biometric Transmission Specification (EBTS) format.

C#

```
public const ushort IncitsTCM1BiometricsEbts = 0x8019;
```

6.2.3.1.4.1.11 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFacelImage Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Face Image Format.

C#

```
public const ushort IncitsTCM1BiometricsFaceImage = 0x0501;
```

6.2.3.1.4.1.12 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerImage Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Finger Image format.

C#

```
public const ushort IncitsTCM1BiometricsFingerImage = 0x0401;
```

6.2.3.1.4.1.13 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerMinutiaeN Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Finger Minutiae format - no extended data.

C#

```
public const ushort IncitsTCM1BiometricsFingerMinutiaeN = 0x0201;
```


6.2.3.1.4.1.14 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerMinutiaeU Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Finger Minutiae format - presence of extended data unspecified.

C#

```
public const ushort IncitsTCM1BiometricsFingerMinutiaeU = 0x0203;
```

6.2.3.1.4.1.15 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerMinutiaeX Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Finger Minutiae Format - with extended data.

C#

```
public const ushort IncitsTCM1BiometricsFingerMinutiaeX = 0x0202;
```

6.2.3.1.4.1.16 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsFingerPatternN Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Finger Pattern format - no extended data.

C#

```
public const ushort IncitsTCM1BiometricsFingerPatternN = 0x0301;
```

6.2.3.1.4.1.17 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsGenericFbiEftsRecord Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Generic FBI EFTS record format.

C#

```
public const ushort IncitsTCM1BiometricsGenericFbiEftsRecord = 0x8014;
```

6.2.3.1.4.1.18 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsGif89aImage Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) CompuServe GIF89a image format.

C#

```
public const ushort IncitsTCM1BiometricsGif89aImage = 0x8006;
```

6.2.3.1.4.1.19 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsHandGeometryN Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Hand Geometry format - no extended data.

C#

```
public const ushort IncitsTCM1BiometricsHandGeometryN = 0x0801;
```

6.2.3.1.4.1.20 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsHandGeometryX Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Hand Geometry format - with extended data.

C#

```
public const ushort IncitsTCM1BiometricsHandGeometryX = 0x0802;
```

6.2.3.1.4.1.21 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsIrisPolar Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Iris Format - polar.

C#

```
public const ushort IncitsTCM1BiometricsIrisPolar = 0x0602;
```

6.2.3.1.4.1.22 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsIrisRectilinear Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Iris Format - rectilinear.

C#

```
public const ushort IncitsTCM1BiometricsIrisRectilinear = 0x0601;
```

6.2.3.1.4.1.23 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsJpeg2000Image Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) JPEG2000 image format.

C#

```
public const ushort IncitsTCM1BiometricsJpeg2000Image = 0x8004;
```

6.2.3.1.4.1.24 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsJpegImage Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) JPEG image format.

C#

```
public const ushort IncitsTCM1BiometricsJpegImage = 0x8003;
```

6.2.3.1.4.1.25 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsMpeg1Part3Media Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) MPEG1 Part 3 media file format.

C#

```
public const ushort IncitsTCM1BiometricsMpeg1Part3Media = 0x800A;
```

6.2.3.1.4.1.26 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsMpegMedia Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) MPEG media file format.

C#

```
public const ushort IncitsTCM1BiometricsMpegMedia = 0x8009;
```

6.2.3.1.4.1.27 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNiem Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) National Information Exchange Model format.

C#

```
public const ushort IncitsTCM1BiometricsNiem = 0x8015;
```

6.2.3.1.4.1.28 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2000Type10Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 10 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2000Type10Record = 0x800E;
```

6.2.3.1.4.1.29 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2000Type13Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL

1-2000 Type 13 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2000Type13Record = 0x800F;
```

6.2.3.1.4.1.30 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2000Type14Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 14 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2000Type14Record = 0x8010;
```

6.2.3.1.4.1.31 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2000Type15Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 15 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2000Type15Record = 0x8011;
```

6.2.3.1.4.1.32 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2000Type16Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 16 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2000Type16Record = 0x8012;
```

6.2.3.1.4.1.33 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2000Type4Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2000 Type 4 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2000Type4Record = 0x800D;
```

6.2.3.1.4.1.34 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2007Type10Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2007 Type 10 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2007Type10Record = 0x8016;
```

6.2.3.1.4.1.35 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2007Type14Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2007 Type 14 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2007Type14Record = 0x8017;
```

6.2.3.1.4.1.36 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_2007Type17Record Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-2007 Type 17 record format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_2007Type17Record = 0x8018;
```

6.2.3.1.4.1.37**CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsNistItl1_200XRecordCollectionForTenPrintCapture Field**

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) NIST-ITL 1-200x record collection for TenPrint Capture format.

C#

```
public const ushort IncitsTCM1BiometricsNistItl1_200XRecordCollectionForTenPrintCapture = 0x8013;
```

6.2.3.1.4.1.38 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsPngImage Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Portable Network Graphics (PNG) data format.

C#

```
public const ushort IncitsTCM1BiometricsPngImage = 0x8007;
```

6.2.3.1.4.1.39 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignCommonFeatureDataN Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - Common feature data - no extended data.

C#

```
public const ushort IncitsTCM1BiometricsSignatureSignCommonFeatureDataN = 0x0703;
```

6.2.3.1.4.1.40 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignCommonFeatureDataX Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - Common feature data - with extended data.

C#

```
public const ushort IncitsTCM1BiometricsSignatureSignCommonFeatureDataX = 0x0704;
```

6.2.3.1.4.1.41**CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataN Field**

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - both Raw and Common feature data - no extended data.

C#

```
public const ushort IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataN = 0x0705;
```

6.2.3.1.4.1.42**CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataX Field**

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - both Raw and Common feature data - with extended data.

C#

```
public const ushort IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataX = 0x0706;
```

6.2.3.1.4.1.43 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignRawDataN Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - Raw data - no extended data.

C#

```
public const ushort IncitsTCM1BiometricsSignatureSignRawDataN = 0x0701;
```

6.2.3.1.4.1.44 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsSignatureSignRawDataX Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) Signature/Sign format - Raw data - with extended data.

C#

```
public const ushort IncitsTCM1BiometricsSignatureSignRawDataX = 0x0702;
```

6.2.3.1.4.1.45 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsTiffImage Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) TIFF image format.

C#

```
public const ushort IncitsTCM1BiometricsTiffImage = 0x8005;
```

6.2.3.1.4.1.46 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsVrml3DObjectData Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) VRML 3D object data format.

C#

```
public const ushort IncitsTCM1BiometricsVrml3DObjectData = 0x800C;
```

6.2.3.1.4.1.47 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsWavAudio Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) WAV audio file format.

C#

```
public const ushort IncitsTCM1BiometricsWavAudio = 0x8008;
```

6.2.3.1.4.1.48 CbeffBdbFormatIdentifiers.IncitsTCM1BiometricsWsqImage Field

The CBEFF DBD format identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639) WSQ compressed fingerprint image format.

C#

```
public const ushort IncitsTCM1BiometricsWsqImage = 0x8001;
```

6.2.3.1.4.1.49**CbeffBdbFormatIdentifiers.IsoIecJtc1SC27ITSecurityTechniquesBiometricReferenceTemplateCertificate Field**

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 27 - IT Security techniques biometric-reference-template-certificate format.

C#

```
public const ushort IsoIecJtc1SC27ITSecurityTechniquesBiometricReferenceTemplateCertificate = 0x0001;
```

6.2.3.1.4.1.50 CbeffBdbFormatIdentifiers.IsoIecJtc1SC37BiometricsFacelImage Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) face-image format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFaceImage = 0x0008;
```

6.2.3.1.4.1.51 CbeffBdbFormatIdentifiers.IsoIecJtc1SC37BiometricsFingerImage Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-image format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerImage = 0x0007;
```

6.2.3.1.4.1.52 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactN Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-compact-n format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeCardCompactN = 0x0006;
```

6.2.3.1.4.1.53 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactRH Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-compact-r-h format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeCardCompactRH = 0x001A;
```

6.2.3.1.4.1.54 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactV Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-compact-v format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeCardCompactV = 0x0005;
```

6.2.3.1.4.1.55 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactVH Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-compact-v-h format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeCardCompactVH = 0x0019;
```

6.2.3.1.4.1.56 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalN Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-normal-n format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeCardNormalN = 0x0004;
```

6.2.3.1.4.1.57 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalRNH Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37-Biometrics (see page 639) finger-minutiae-card-normal-r-nh format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeCardNormalRNH = 0x001C;
```

6.2.3.1.4.1.58 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalV Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-card-normal-v format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeCardNormalV = 0x0003;
```

6.2.3.1.4.1.59 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalVNH Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639)

finger-minutiae-card-normal-v-nh format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeCardNormalVNH = 0x001B;
```

6.2.3.1.4.1.60 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeRecordN Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-record-n format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeRecordN = 0x0001;
```

6.2.3.1.4.1.61 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerMinutiaeRecordX Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-minutiae-record-x format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerMinutiaeRecordX = 0x0002;
```

6.2.3.1.4.1.62

CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSkeletalDataCardCompact Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-skeletal-data-card-compact format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerPatternSkeletalDataCardCompact = 0x0013;
```

6.2.3.1.4.1.63 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSkeletalDataCardNormal Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-skeletal-data-card-normal format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerPatternSkeletalDataCardNormal = 0x0012;
```

6.2.3.1.4.1.64 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSkeletalDataRecord Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-skeletal-data-record format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerPatternSkeletalDataRecord = 0x0011;
```

6.2.3.1.4.1.65

CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSpectralDiscreteFourierTransform Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-spectral-discrete-fourier-transform format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerPatternSpectralDiscreteFourierTransform = 0x000C;
```

6.2.3.1.4.1.66 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSpectralGaborFilter Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-spectral-gabor-filter format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsFingerPatternSpectralGaborFilter = 0x000D;
```

6.2.3.1.4.1.67

CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsFingerPatternSpectralQuantizedCOSinusoidalTriplet Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) finger-pattern-spectral-quantized-co-sinusoidal-triplet format.

C#

```
public const ushort  
IsoIecJtc1SC37BiometricsFingerPatternSpectralQuantizedCOSinusoidalTriplet = 0x000A;
```

6.2.3.1.4.1.68 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsHandGeometrySilhouette Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) hand-geometry-silhouette format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsHandGeometrySilhouette = 0x0018;
```

6.2.3.1.4.1.69 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsIrisImagePolar Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) iris-image-polar format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsIrisImagePolar = 0x000B;
```

6.2.3.1.4.1.70 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsIrisImageRectilinear Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) iris-image-rectilinear format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsIrisImageRectilinear = 0x0009;
```

6.2.3.1.4.1.71 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsSignatureSignTimeSeriesCompact Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) signature-sign-time-series-compact format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsSignatureSignTimeSeriesCompact = 0x000F;
```

6.2.3.1.4.1.72 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsSignatureSignTimeSeriesFull Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) signature-sign-time-series-full format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsSignatureSignTimeSeriesFull = 0x000E;
```

6.2.3.1.4.1.73 CbeffBdbFormatIdentifiers.IsolecJtc1SC37BiometricsVascularImageData Field

The CBEFF DBD format identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639) vascular-image-data format.

C#

```
public const ushort IsoIecJtc1SC37BiometricsVascularImageData = 0x0014;
```


6.2.3.1.4.1.74 CbeffBdbFormatIdentifiers.NeurotechnologijaNLRecord3 Field

The CBEFF DBD format identifier assigned by Neurotechnologija for NLRecord version 3 format.

C#

```
public const ushort NeurotechnologijaNLRecord3 = 0x0302;
```

6.2.3.1.4.1.75 CbeffBdbFormatIdentifiers.NeurotechnologijaNTemplate Field

The CBEFF DBD format identifier assigned by Neurotechnologija for NTemplate (see page 726) format.

C#

```
public const ushort NeurotechnologijaNTemplate = 0x0000;
```

6.2.3.1.5 CbeffBiometricOrganizations Class

Specifies CBEFF biometric organizations identifiers registered with International Biometric Industry Association (IBIA).

Class Hierarchy

C#

```
public static class CbeffBiometricOrganizations;
```

Remarks

To use the functionality of this class you should include Neurotec.Biometrics.Cbeff.dll into your project.

CbeffBiometricOrganizations Fields

	Name	Description
	_123ID (see page 779)	The CBEFF biometric organization identifier assigned to 123ID, Inc.
	A4Vision (see page 779)	The CBEFF biometric organization identifier assigned to A 4 Vision, S.A.
	AcSysBiometrics (see page 779)	The CBEFF biometric organization identifier assigned to AcSys Biometrics (see page 639) Corp.
	ActivCard (see page 779)	The CBEFF biometric organization identifier assigned to ActivCard, Inc.
	AntheusTechnology (see page 779)	The CBEFF biometric organization identifier assigned to Antheus Technology.
	AOptix (see page 779)	The CBEFF biometric organization identifier assigned to AOptix.
	Atmel (see page 779)	The CBEFF biometric organization identifier assigned to Atmel.
	AuroraWirelessTechnologies (see page 780)	The CBEFF biometric organization identifier assigned to Aurora Wireless Technologies, Inc.
	AuthenTec (see page 780)	The CBEFF biometric organization identifier assigned to Authen Tec.
	Aware (see page 780)	The CBEFF biometric organization identifier assigned to Aware, Inc.
	BioKeyInternational (see page 780)	The CBEFF biometric organization identifier assigned to BIO-key International.
	BioKeyInternational2 (see page 780)	The CBEFF biometric organization identifier assigned to Bio-Key International.
	BioLinkTechnologiesInternational (see page 780)	The CBEFF biometric organization identifier assigned to BioLink Technologies International, Inc.
	BioLogicaSistemas (see page 780)	The CBEFF biometric organization identifier assigned to BioLogica Sistemas Ltda.
	Bioscrypt (see page 780)	The CBEFF biometric organization identifier assigned to Bioscrypt, Inc.
	Biovision (see page 780)	The CBEFF biometric organization identifier assigned to Biovision, Inc.

◆	BioWise (see page 781)	The CBEFF biometric organization identifier assigned to Bio Wise, NV.
◆	CyberSign (see page 781)	The CBEFF biometric organization identifier assigned to Cyber SIGN, Inc.
◆	CogentSystems (see page 781)	The CBEFF biometric organization identifier assigned to Cogent Systems, Inc.
◆	CryptoMetrics (see page 781)	The CBEFF biometric organization identifier assigned to CryptoMetrics.
◆	CrossMatchTechnologies (see page 781)	The CBEFF biometric organization identifier assigned to Cross Match Technologies, Inc.
◆	DermalogIdentificationSystems (see page 781)	The CBEFF biometric organization identifier assigned to DERMALOG Identification Systems, GmbH.
◆	Digent (see page 781)	The CBEFF biometric organization identifier assigned to Digent Co., Ltd.
◆	DigitalPersona (see page 781)	The CBEFF biometric organization identifier assigned to DigitalPersona, Inc.
◆	EasternGoldenFingerTechnologyBeijing (see page 781)	The CBEFF biometric organization identifier assigned to Eastern Golden Finger Technology Beijing Co., Ltd.
◆	EastShoreTechnologies (see page 782)	The CBEFF biometric organization identifier assigned to East Shore Technologies.
◆	ECryp (see page 782)	The CBEFF biometric organization identifier assigned to eCryp, Inc.
◆	FederalOfficeForInformationSecurity (see page 782)	The CBEFF biometric organization identifier assigned to Federal Office for Information Security (BSI).
◆	FingerprintCards (see page 782)	The CBEFF biometric organization identifier assigned to Fingerprint Cards AB.
◆	Fujitsu (see page 782)	The CBEFF biometric organization identifier assigned to Fujitsu Ltd.
◆	GermanInstituteForStandardization (see page 782)	The CBEFF biometric organization identifier assigned to German Institute for Standardization (Deutsches Institut für Normung E.V.-DIN).
◆	GreenBitAmericas (see page 782)	The CBEFF biometric organization identifier assigned to Green Bit Americas Inc.
◆	GriauleTecnologia (see page 782)	The CBEFF biometric organization identifier assigned to Griaule Tecnologia LTDA.
◆	GuardwareSystems (see page 782)	The CBEFF biometric organization identifier assigned to Guardware Systems, Ltd.
◆	Hitachi (see page 783)	The CBEFF biometric organization identifier assigned to Hitachi, LTD.
◆	HumanScan (see page 783)	The CBEFF biometric organization identifier assigned to HumanScan GmbH.
◆	Id3Semiconductors (see page 783)	The CBEFF biometric organization identifier assigned to id3 Semiconductors.
◆	Identix (see page 783)	The CBEFF biometric organization identifier assigned to Identix Corporation.
◆	Identix2 (see page 783)	The CBEFF biometric organization identifier assigned to Identix (see page 783), Inc.
◆	ImageWareSystems (see page 783)	The CBEFF biometric organization identifier assigned to Image Ware Systems, Inc.
◆	IncitsTCM1Biometrics (see page 783)	The CBEFF biometric organization identifier assigned to INCITS Technical Committee M1 - Biometrics (see page 639).
◆	InfineonTechnologies (see page 783)	The CBEFF biometric organization identifier assigned to Infineon Technologies, AG.
◆	Innovatrics (see page 783)	The CBEFF biometric organization identifier assigned to Innovatrics.
◆	InternationalBiometricGroup (see page 784)	The CBEFF biometric organization identifier assigned to International Biometric Group.

InvestorsHelpers (see page 784)	The CBEFF biometric organization identifier assigned to InvestorsHelpers, Inc.
IridianTechnologies (see page 784)	The CBEFF biometric organization identifier assigned to Iridian Technologies, Inc.
IsolecJtc1SC27ITSecurityTechniques (see page 784)	The CBEFF biometric organization identifier assigned to ISO/IEC JTC 1 SC 27 - IT Security techniques.
IsolecJtc1SC37Biometrics (see page 784)	The CBEFF biometric organization identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639).
JanusAssociates (see page 784)	The CBEFF biometric organization identifier assigned to JANUS Associates, Inc.
LGElectronicsUsa (see page 784)	The CBEFF biometric organization identifier assigned to LG Electronics USA, Inc.
LogicoSmartcardSolutions (see page 784)	The CBEFF biometric organization identifier assigned to LOGICO Smartcard Solutions, AG.
Lumidigm (see page 784)	The CBEFF biometric organization identifier assigned to Lumidigm.
NecSolutionsAmerica (see page 785)	The CBEFF biometric organization identifier assigned to NEC Solutions America, Inc.
Neurotechnologija (see page 785)	The CBEFF biometric organization identifier assigned to Neurotechnologija.
Nist (see page 785)	The CBEFF biometric organization identifier assigned to National Institute of Standards (see page 753) and Technology (NIST).
Nitgen (see page 785)	The CBEFF biometric organization identifier assigned to NITGEN.
NotForUse (see page 785)	The CBEFF biometric organization identifier which is not allowed to be used by any of the biometric organizations.
OmniMotorola (see page 785)	The CBEFF biometric organization identifier assigned to Motorola.
OmniPerception (see page 785)	The CBEFF biometric organization identifier assigned to OmniPerception Limited.
PreciseBiometrics (see page 785)	The CBEFF biometric organization identifier assigned to Precise Biometrics (see page 639), AB.
Private (see page 785)	The CBEFF biometric organization identifier for private use (not uniquely assigned by IBIA).
RecognitionSystems (see page 786)	The CBEFF biometric organization identifier assigned to Recognition Systems, Inc.
Saflink (see page 786)	The CBEFF biometric organization identifier assigned to SAFLINK Corporation.
SagemMorpho (see page 786)	The CBEFF biometric organization identifier assigned to Sagem Morpho.
SecuGen (see page 786)	The CBEFF biometric organization identifier assigned to SecuGen Corporation.
SecureDesign (see page 786)	The CBEFF biometric organization identifier assigned to Secure Design.
SilexTechnology (see page 786)	The CBEFF biometric organization identifier assigned to Silex Technology, Inc.
SondaTechnologies (see page 786)	The CBEFF biometric organization identifier assigned to Sonda Technologies.
StarTekEngineering (see page 786)	The CBEFF biometric organization identifier assigned to StarTek Engineering, Inc.
Suprema (see page 786)	The CBEFF biometric organization identifier assigned to Suprema, Inc.
Testing1 (see page 787)	The CBEFF biometric organization identifier reserved for testing.
Testing2 (see page 787)	The CBEFF biometric organization identifier reserved for testing.
ThalesIdenitfication (see page 787)	The CBEFF biometric organization identifier assigned to Thales Idenitfication.

	UltraScan ( see page 787)	The CBEFF biometric organization identifier assigned to Ultra-Scan Corporation.
	Upek ( see page 787)	The CBEFF biometric organization identifier assigned to UPEK, Inc.
	Validity ( see page 787)	The CBEFF biometric organization identifier assigned to Validity, Inc.
	Veridicom ( see page 787)	The CBEFF biometric organization identifier assigned to Veridicom, Inc.
	Veridt ( see page 787)	The CBEFF biometric organization identifier assigned to Veridt.
	Viisage ( see page 787)	The CBEFF biometric organization identifier assigned to Viisage.
	WarwickWarp ( see page 788)	The CBEFF biometric organization identifier assigned to Warwick Warp Ltd.
	XTec ( see page 788)	The CBEFF biometric organization identifier assigned to XTec, Incorporated.

6.2.3.1.5.1 CbeffBiometricOrganizations Fields

6.2.3.1.5.1.1 CbeffBiometricOrganizations._123ID Field

The CBEFF biometric organization identifier assigned to 123ID, Inc.

C#

```
public const ushort _123ID = 0x0038;
```

6.2.3.1.5.1.2 CbeffBiometricOrganizations.A4Vision Field

The CBEFF biometric organization identifier assigned to A 4 Vision, S.A.

C#

```
public const ushort A4Vision = 0x0010;
```

6.2.3.1.5.1.3 CbeffBiometricOrganizations.AcSysBiometrics Field

The CBEFF biometric organization identifier assigned to AcSys Biometrics ( see page 639) Corp.

C#

```
public const ushort AcSysBiometrics = 0x0022;
```

6.2.3.1.5.1.4 CbeffBiometricOrganizations.ActivCard Field

The CBEFF biometric organization identifier assigned to ActivCard, Inc.

C#

```
public const ushort ActivCard = 0x0020;
```

6.2.3.1.5.1.5 CbeffBiometricOrganizations.AntheusTechnology Field

The CBEFF biometric organization identifier assigned to Antheus Technology.

C#

```
public const ushort AntheusTechnology = 0x0032;
```

6.2.3.1.5.1.6 CbeffBiometricOrganizations.AOptix Field

The CBEFF biometric organization identifier assigned to AOptix.

C#

```
public const ushort AOptix = 0x0047;
```

6.2.3.1.5.1.7 CbeffBiometricOrganizations.Atmel Field

The CBEFF biometric organization identifier assigned to Atmel.

C#

```
public const ushort Atmel = 0x0041;
```

6.2.3.1.5.1.8 CbeffBiometricOrganizations.AuroraWirelessTechnologies Field

The CBEFF biometric organization identifier assigned to Aurora Wireless Technologies, Inc.

C#

```
public const ushort AuroraWirelessTechnologies = 0x0014;
```

6.2.3.1.5.1.9 CbeffBiometricOrganizations.AuthenTec Field

The CBEFF biometric organization identifier assigned to Authen Tec.

C#

```
public const ushort AuthenTec = 0x0042;
```

6.2.3.1.5.1.10 CbeffBiometricOrganizations.Aware Field

The CBEFF biometric organization identifier assigned to Aware, Inc.

C#

```
public const ushort Aware = 0x003B;
```

6.2.3.1.5.1.11 CbeffBiometricOrganizations.BioKeyInternational Field

The CBEFF biometric organization identifier assigned to BIO-key International.

C#

```
public const ushort BioKeyInternational = 0x0030;
```

6.2.3.1.5.1.12 CbeffBiometricOrganizations.BioKeyInternational2 Field

The CBEFF biometric organization identifier assigned to Bio-Key International.

C#

```
public const ushort BioKeyInternational2 = 0x0039;
```

6.2.3.1.5.1.13 CbeffBiometricOrganizations.BioLinkTechnologiesInternational Field

The CBEFF biometric organization identifier assigned to BioLink Technologies International, Inc.

C#

```
public const ushort BioLinkTechnologiesInternational = 0x001F;
```

6.2.3.1.5.1.14 CbeffBiometricOrganizations.BioLogicaSistemas Field

The CBEFF biometric organization identifier assigned to BioLogica Sistemas Ltda.

C#

```
public const ushort BioLogicaSistemas = 0x004C;
```

6.2.3.1.5.1.15 CbeffBiometricOrganizations.Bioscrypt Field

The CBEFF biometric organization identifier assigned to Bioscrypt, Inc.

C#

```
public const ushort Bioscrypt = 0x0002;
```

6.2.3.1.5.1.16 CbeffBiometricOrganizations.Biovision Field

The CBEFF biometric organization identifier assigned to Biovision, Inc.

C#

```
public const ushort Biovision = 0x0045;
```

6.2.3.1.5.1.17 CbeffBiometricOrganizations.BioWise Field

The CBEFF biometric organization identifier assigned to Bio Wise, NV.

C#

```
public const ushort BioWise = 0x0024;
```

6.2.3.1.5.1.18 CbeffBiometricOrganizations.CyberSign Field

The CBEFF biometric organization identifier assigned to Cyber SIGN, Inc.

C#

```
public const ushort CyberSign = 0x0007;
```

6.2.3.1.5.1.19 CbeffBiometricOrganizations.CogentSystems Field

The CBEFF biometric organization identifier assigned to Cogent Systems, Inc.

C#

```
public const ushort CogentSystems = 0x0017;
```

6.2.3.1.5.1.20 CbeffBiometricOrganizations.CryptoMetrics Field

The CBEFF biometric organization identifier assigned to CryptoMetrics.

C#

```
public const ushort CryptoMetrics = 0x002F;
```

6.2.3.1.5.1.21 CbeffBiometricOrganizations.CrossMatchTechnologies Field

The CBEFF biometric organization identifier assigned to Cross Match Technologies, Inc.

C#

```
public const ushort CrossMatchTechnologies = 0x0018;
```

6.2.3.1.5.1.22 CbeffBiometricOrganizations.DermalogIdentificationSystems Field

The CBEFF biometric organization identifier assigned to DERMALOG Identification Systems, GmbH.

C#

```
public const ushort DermalogIdentificationSystems = 0x000D;
```

6.2.3.1.5.1.23 CbeffBiometricOrganizations.Digent Field

The CBEFF biometric organization identifier assigned to Digent Co., Ltd.

C#

```
public const ushort Digent = 0x0048;
```

6.2.3.1.5.1.24 CbeffBiometricOrganizations.DigitalPersona Field

The CBEFF biometric organization identifier assigned to DigitalPersona, Inc.

C#

```
public const ushort DigitalPersona = 0x0033;
```

6.2.3.1.5.1.25 CbeffBiometricOrganizations.EasternGoldenFingerTechnologyBeijing Field

The CBEFF biometric organization identifier assigned to Eastern Golden Finger Technology Beijing Co., Ltd.

C#

```
public const ushort EasternGoldenFingerTechnologyBeijing = 0x004A;
```

6.2.3.1.5.1.26 CbeffBiometricOrganizations.EastShoreTechnologies Field

The CBEFF biometric organization identifier assigned to East Shore Technologies.

C#

```
public const ushort EastShoreTechnologies = 0x0036;
```

6.2.3.1.5.1.27 CbeffBiometricOrganizations.ECryp Field

The CBEFF biometric organization identifier assigned to eCryp, Inc.

C#

```
public const ushort ECryp = 0x0008;
```

6.2.3.1.5.1.28 CbeffBiometricOrganizations.FederalOfficeForInformationSecurity Field

The CBEFF biometric organization identifier assigned to Federal Office for Information Security (BSI).

C#

```
public const ushort FederalOfficeForInformationSecurity = 0x004B;
```

6.2.3.1.5.1.29 CbeffBiometricOrganizations.FingerprintCards Field

The CBEFF biometric organization identifier assigned to Fingerprint Cards AB.

C#

```
public const ushort FingerprintCards = 0x0009;
```

6.2.3.1.5.1.30 CbeffBiometricOrganizations.Fujitsu Field

The CBEFF biometric organization identifier assigned to Fujitsu Ltd.

C#

```
public const ushort Fujitsu = 0x0046;
```

6.2.3.1.5.1.31 CbeffBiometricOrganizations.GermanInstituteForStandardization Field

The CBEFF biometric organization identifier assigned to German Institute for Standardization (Deutsches Institut für Normung E.V.-DIN).

C#

```
public const ushort GermanInstituteForStandardization = 0x001A;
```

6.2.3.1.5.1.32 CbeffBiometricOrganizations.GreenBitAmericas Field

The CBEFF biometric organization identifier assigned to Green Bit Americas Inc.

C#

```
public const ushort GreenBitAmericas = 0x0040;
```

6.2.3.1.5.1.33 CbeffBiometricOrganizations.GriauleTecnologia Field

The CBEFF biometric organization identifier assigned to Griaule Tecnologia LTDA.

C#

```
public const ushort GriauleTecnologia = 0x003A;
```

6.2.3.1.5.1.34 CbeffBiometricOrganizations.GuardwareSystems Field

The CBEFF biometric organization identifier assigned to Guardware Systems, Ltd.

C#

```
public const ushort GuardwareSystems = 0x0026;
```

6.2.3.1.5.1.35 CbeffBiometricOrganizations.Hitachi Field

The CBEFF biometric organization identifier assigned to Hitachi, LTD.

C#

```
public const ushort Hitachi = 0x0037;
```

6.2.3.1.5.1.36 CbeffBiometricOrganizations.HumanScan Field

The CBEFF biometric organization identifier assigned to HumanScan GmbH.

C#

```
public const ushort HumanScan = 0x0021;
```

6.2.3.1.5.1.37 CbeffBiometricOrganizations.Id3Semiconductors Field

The CBEFF biometric organization identifier assigned to id3 Semiconductors.

C#

```
public const ushort Id3Semiconductors = 0x003F;
```

6.2.3.1.5.1.38 CbeffBiometricOrganizations.Identix Field

The CBEFF biometric organization identifier assigned to Identix Corporation.

C#

```
public const ushort Identix = 0x0003;
```

6.2.3.1.5.1.39 CbeffBiometricOrganizations.Identix2 Field

The CBEFF biometric organization identifier assigned to Identix ( see page 783), Inc.

C#

```
public const ushort Identix2 = 0x000C;
```

6.2.3.1.5.1.40 CbeffBiometricOrganizations.ImageWareSystems Field

The CBEFF biometric organization identifier assigned to Image Ware Systems, Inc.

C#

```
public const ushort ImageWareSystems = 0x0043;
```

6.2.3.1.5.1.41 CbeffBiometricOrganizations.IncitsTCM1Biometrics Field

The CBEFF biometric organization identifier assigned to INCITS Technical Committee M1 - Biometrics ( see page 639).

C#

```
public const ushort IncitsTCM1Biometrics = 0x001B;
```

6.2.3.1.5.1.42 CbeffBiometricOrganizations.InfineonTechnologies Field

The CBEFF biometric organization identifier assigned to Infineon Technologies, AG.

C#

```
public const ushort InfineonTechnologies = 0x0004;
```

6.2.3.1.5.1.43 CbeffBiometricOrganizations.Innovatrics Field

The CBEFF biometric organization identifier assigned to Innovatrics.

C#

```
public const ushort Innovatrics = 0x0035;
```

6.2.3.1.5.1.44 CbeffBiometricOrganizations.InternationalBiometricGroup Field

The CBEFF biometric organization identifier assigned to International Biometric Group.

C#

```
public const ushort InternationalBiometricGroup = 0x0016;
```

6.2.3.1.5.1.45 CbeffBiometricOrganizations.InvestorsHelpers Field

The CBEFF biometric organization identifier assigned to InvestorsHelpers, Inc.

C#

```
public const ushort InvestorsHelpers = 0x0027;
```

6.2.3.1.5.1.46 CbeffBiometricOrganizations.IridianTechnologies Field

The CBEFF biometric organization identifier assigned to Iridian Technologies, Inc.

C#

```
public const ushort IridianTechnologies = 0x0005;
```

6.2.3.1.5.1.47 CbeffBiometricOrganizations.IsoIecJtc1SC27ITSecurityTechniques Field

The CBEFF biometric organization identifier assigned to ISO/IEC JTC 1 SC 27 - IT Security techniques.

C#

```
public const ushort IsoIecJtc1SC27ITSecurityTechniques = 0x0102;
```

6.2.3.1.5.1.48 CbeffBiometricOrganizations.IsoIecJtc1SC37Biometrics Field

The CBEFF biometric organization identifier assigned to ISO/IEC JTC 1 SC 37 - Biometrics (see page 639).

C#

```
public const ushort IsoIecJtc1SC37Biometrics = 0x0101;
```

6.2.3.1.5.1.49 CbeffBiometricOrganizations.JanusAssociates Field

The CBEFF biometric organization identifier assigned to JANUS Associates, Inc.

C#

```
public const ushort JanusAssociates = 0x002C;
```

6.2.3.1.5.1.50 CbeffBiometricOrganizations.LGElectronicsUsa Field

The CBEFF biometric organization identifier assigned to LG Electronics USA, Inc.

C#

```
public const ushort LGElectronicsUsa = 0x002A;
```

6.2.3.1.5.1.51 CbeffBiometricOrganizations.LogicoSmartcardSolutions Field

The CBEFF biometric organization identifier assigned to LOGICO Smartcard Solutions, AG.

C#

```
public const ushort LogicoSmartcardSolutions = 0x000E;
```

6.2.3.1.5.1.52 CbeffBiometricOrganizations.Lumidigm Field

The CBEFF biometric organization identifier assigned to Lumidigm.

C#

```
public const ushort Lumidigm = 0x0025;
```

6.2.3.1.5.1.53 CbeffBiometricOrganizations.NecSolutionsAmerica Field

The CBEFF biometric organization identifier assigned to NEC Solutions America, Inc.

C#

```
public const ushort NecSolutionsAmerica = 0x0011;
```

6.2.3.1.5.1.54 CbeffBiometricOrganizations.Neurotechnologija Field

The CBEFF biometric organization identifier assigned to Neurotechnologija.

C#

```
public const ushort Neurotechnologija = 0x0031;
```

6.2.3.1.5.1.55 CbeffBiometricOrganizations.Nist Field

The CBEFF biometric organization identifier assigned to National Institute of Standards (see page 753) and Technology (NIST).

C#

```
public const ushort Nist = 0x000F;
```

6.2.3.1.5.1.56 CbeffBiometricOrganizations.Nitgen Field

The CBEFF biometric organization identifier assigned to NITGEN.

C#

```
public const ushort Nitgen = 0x001C;
```

6.2.3.1.5.1.57 CbeffBiometricOrganizations.NotForUse Field

The CBEFF biometric organization identifier which is not allowed to be used by any of the biometric organizations.

C#

```
public const ushort NotForUse = 0xFFFF;
```

6.2.3.1.5.1.58 CbeffBiometricOrganizations.OmniMotorola Field

The CBEFF biometric organization identifier assigned to Motorola.

C#

```
public const ushort OmniMotorola = 0x002E;
```

6.2.3.1.5.1.59 CbeffBiometricOrganizations.OmniPerception Field

The CBEFF biometric organization identifier assigned to OmniPerception Limited.

C#

```
public const ushort OmniPerception = 0x002D;
```

6.2.3.1.5.1.60 CbeffBiometricOrganizations.PreciseBiometrics Field

The CBEFF biometric organization identifier assigned to Precise Biometrics (see page 639), AB.

C#

```
public const ushort PreciseBiometrics = 0x000B;
```

6.2.3.1.5.1.61 CbeffBiometricOrganizations.Private Field

The CBEFF biometric organization identifier for private use (not uniquely assigned by IBIA).

C#

```
public const ushort Private = 0xFEFE;
```

6.2.3.1.5.1.62 CbeffBiometricOrganizations.RecognitionSystems Field

The CBEFF biometric organization identifier assigned to Recognition Systems, Inc.

C#

```
public const ushort RecognitionSystems = 0x0019;
```

6.2.3.1.5.1.63 CbeffBiometricOrganizations.Saflink Field

The CBEFF biometric organization identifier assigned to SAFLINK Corporation.

C#

```
public const ushort Saflink = 0x0001;
```

6.2.3.1.5.1.64 CbeffBiometricOrganizations.SagemMorpho Field

The CBEFF biometric organization identifier assigned to Sagem Morpho.

C#

```
public const ushort SagemMorpho = 0x001D;
```

6.2.3.1.5.1.65 CbeffBiometricOrganizations.SecuGen Field

The CBEFF biometric organization identifier assigned to SecuGen Corporation.

C#

```
public const ushort SecuGen = 0x000A;
```

6.2.3.1.5.1.66 CbeffBiometricOrganizations.SecureDesign Field

The CBEFF biometric organization identifier assigned to Secure Design.

C#

```
public const ushort SecureDesign = 0x003D;
```

6.2.3.1.5.1.67 CbeffBiometricOrganizations.SilexTechnology Field

The CBEFF biometric organization identifier assigned to Silex Technology, Inc.

C#

```
public const ushort SilexTechnology = 0x0023;
```

6.2.3.1.5.1.68 CbeffBiometricOrganizations.SondaTechnologies Field

The CBEFF biometric organization identifier assigned to Sonda Technologies.

C#

```
public const ushort SondaTechnologies = 0x003C;
```

6.2.3.1.5.1.69 CbeffBiometricOrganizations.StarTekEngineering Field

The CBEFF biometric organization identifier assigned to StarTek Engineering, Inc.

C#

```
public const ushort StarTekEngineering = 0x002B;
```

6.2.3.1.5.1.70 CbeffBiometricOrganizations.Suprema Field

The CBEFF biometric organization identifier assigned to Suprema, Inc.

C#

```
public const ushort Suprema = 0x0044;
```

6.2.3.1.5.1.71 CbeffBiometricOrganizations.Testing1 Field

The CBEFF biometric organization identifier reserved for testing.

C#

```
public const ushort Testing1 = 0xFFF0;
```

6.2.3.1.5.1.72 CbeffBiometricOrganizations.Testing2 Field

The CBEFF biometric organization identifier reserved for testing.

C#

```
public const ushort Testing2 = 0xFFE;
```

6.2.3.1.5.1.73 CbeffBiometricOrganizations.ThalesIdenitfication Field

The CBEFF biometric organization identifier assigned to Thales Identification.

C#

```
public const ushort ThalesIdenitfication = 0x0015;
```

6.2.3.1.5.1.74 CbeffBiometricOrganizations.UltraScan Field

The CBEFF biometric organization identifier assigned to Ultra-Scan Corporation.

C#

```
public const ushort UltraScan = 0x0013;
```

6.2.3.1.5.1.75 CbeffBiometricOrganizations.Upek Field

The CBEFF biometric organization identifier assigned to UPEK, Inc.

C#

```
public const ushort Upek = 0x0012;
```

6.2.3.1.5.1.76 CbeffBiometricOrganizations.Validity Field

The CBEFF biometric organization identifier assigned to Validity, Inc.

C#

```
public const ushort Validity = 0x0028;
```

6.2.3.1.5.1.77 CbeffBiometricOrganizations.Veridicom Field

The CBEFF biometric organization identifier assigned to Veridicom, Inc.

C#

```
public const ushort Veridicom = 0x0006;
```

6.2.3.1.5.1.78 CbeffBiometricOrganizations.Veridt Field

The CBEFF biometric organization identifier assigned to Veridt.

C#

```
public const ushort Veridt = 0x003E;
```

6.2.3.1.5.1.79 CbeffBiometricOrganizations.Viisage Field

The CBEFF biometric organization identifier assigned to Viisage.

C#

```
public const ushort Viisage = 0x0029;
```

6.2.3.1.5.1.80 CbeffBiometricOrganizations.WarwickWarp Field

The CBEFF biometric organization identifier assigned to Warwick Warp Ltd.

C#

```
public const ushort WarwickWarp = 0x0049;
```

6.2.3.1.5.1.81 CbeffBiometricOrganizations.XTec Field

The CBEFF biometric organization identifier assigned to XTec, Incorporated.

C#

```
public const ushort XTec = 0x0034;
```

6.2.3.1.6 FMCard Class

Provides functionality for converting ISO/IEC 19794-2:2005 card format standard templates to and from Neurotechnology Finger Records (NFRecords).

Class Hierarchy

C#

```
public static class FMCard;
```

FMCard Fields

	Name	Description
	DefaultMaxEnrollMC (see page 788)	Default maximal minutia count for enrollment.
	DefaultMaxVerifyMC (see page 789)	Default maximal minutia count for verification.
	DefaultMinEnrollMC (see page 789)	Default minimal minutia count for enrollment.
	DefaultMinVerifyMC (see page 789)	Default maximal minutia count for verification.

FMCard Methods

	Name	Description
	ReadCompactSizeMinutiae (see page 789)	Reads compact size ISO/IEC 19794-2:2005 card format standard template from memory buffer and converts it to NFRecord (see page 666).
	ReadNormalSizeMinutiae (see page 791)	Reads normal size ISO/IEC 19794-2:2005 card format standard template from memory buffer and converts it to NFRecord (see page 666).
	WriteCompactSizeMinutiae (see page 793)	Converts the NFRecord (see page 666) to compact size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.
	WriteNormalSizeMinutiae (see page 797)	Converts NFRecord (see page 666) to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

6.2.3.1.6.1 FMCard Fields

6.2.3.1.6.1.1 FMCard.DefaultMaxEnrollMC Field

Default maximal minutia count for enrollment.

C#

```
public const int DefaultMaxEnrollMC = 60;
```

6.2.3.1.6.1.2 FMCARD.DefaultMaxVerifyMC Field

Default maximal minutia count for verification.

C#

```
public const int DefaultMaxVerifyMC = 60;
```

6.2.3.1.6.1.3 FMCARD.DefaultMinEnrollMC Field

Default minimal minutia count for enrollment.

C#

```
public const int DefaultMinEnrollMC = 16;
```

6.2.3.1.6.1.4 FMCARD.DefaultMinVerifyMC Field

Default maximal minutia count for verification.

C#

```
public const int DefaultMinVerifyMC = 12;
```

6.2.3.1.6.2 FMCARD Methods

6.2.3.1.6.2.1 ReadCompactSizeMinutiae Method

6.2.3.1.6.2.1.1 FMCARD.ReadCompactSizeMinutiae Method (IntPtr, int, FmcMinutiaOrder)

Reads compact size ISO/IEC 19794-2:2005 card format standard template from memory buffer and converts it to NFRecord (see page 666).

C#

```
public static NFRecord ReadCompactSizeMinutiae(IntPtr pBuffer, int bufferSize,
FmcMinutiaOrder order);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains minutiae.
int bufferSize	Size of memory buffer that contains minutiae.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.

Returns

The NFRecord (see page 666) object.

6.2.3.1.6.2.1.2 FMCARD.ReadCompactSizeMinutiae Method (IntPtr, int, FmcMinutiaOrder, uint)

Reads compact size ISO/IEC 19794-2:2005 card format standard template from memory buffer and converts it to NFRecord (see page 666).

C#

```
public static NFRecord ReadCompactSizeMinutiae(IntPtr pBuffer, int bufferSize,
FmcMinutiaOrder order, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains minutiae.

int bufferSize	Size of memory buffer that contains minutiae.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
uint flags	Reserved. Must be zero.

Returns

The NfRecord (see page 666) object.

6.2.3.1.6.2.1.3 FMCARD.ReadCompactSizeMinutiae Method (byte[], FmcMinutiaOrder)

Reads compact size ISO/IEC 19794-2:2005 card format standard template from byte array and converts it to NfRecord (see page 666).

C#

```
public static NfRecord ReadCompactSizeMinutiae(byte[] buffer, FmcMinutiaOrder order);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains minutiae.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.

Returns

The NfRecord (see page 666) object.

6.2.3.1.6.2.1.4 FMCARD.ReadCompactSizeMinutiae Method (byte[], FmcMinutiaOrder, uint)

Reads compact size ISO/IEC 19794-2:2005 card format standard template from byte array and converts it to NfRecord (see page 666).

C#

```
public static NfRecord ReadCompactSizeMinutiae(byte[] buffer, FmcMinutiaOrder order, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains minutiae.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
uint flags	Reserved. Must be zero.

Returns

The NfRecord (see page 666) object.

6.2.3.1.6.2.1.5 FMCARD.ReadCompactSizeMinutiae Method (byte[], int, int, FmcMinutiaOrder)

Reads compact size ISO/IEC 19794-2:2005 card format standard template from byte array and converts it to NfRecord (see page 666).

C#

```
public static NfRecord ReadCompactSizeMinutiae(byte[] buffer, int index, int count, FmcMinutiaOrder order);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains minutiae.
int index	Start index of byte array.
int count	The length of byte array.

FmcMinutiaOrder order	Order in which minutiae are in the byte array.

Returns

The NfRecord (see page 666) object.

6.2.3.1.6.2.1.6 FMCARD.ReadCompactSizeMinutiae Method (byte[], int, int, FmcMinutiaOrder, uint)

Reads compact size ISO/IEC 19794-2:2005 card format standard template from byte array and converts it to NfRecord (see page 666).

C#

```
public static NfRecord ReadCompactSizeMinutiae(byte[] buffer, int index, int count,
FmcMinutiaOrder order, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains minutiae.
int index	Start index of byte array.
int count	The length of byte array.
FmcMinutiaOrder order	Order in which minutiae are in the byte array.
uint flags	Reserved. Must be zero.

Returns

The NfRecord (see page 666) object.

Exceptions

Exceptions	Description
ArgumentOutOfRangeException	index is less than zero. - or - count is less than zero.
ArgumentException	index plus count is greater than buffer length

6.2.3.1.6.2.2 ReadNormalSizeMinutiae Method**6.2.3.1.6.2.2.1 FMCARD.ReadNormalSizeMinutiae Method (IntPtr, int, FmcMinutiaOrder)**

Reads normal size ISO/IEC 19794-2:2005 card format standard template from memory buffer and converts it to NfRecord (see page 666).

C#

```
public static NfRecord ReadNormalSizeMinutiae(IntPtr pBuffer, int bufferSize,
FmcMinutiaOrder order);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains minutiae.
int bufferSize	Size of memory buffer that contains minutiae.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.

Returns

The NfRecord (see page 666) object.

6.2.3.1.6.2.2.2 FMCARD.ReadNormalSizeMinutiae Method (IntPtr, int, FmcMinutiaOrder, uint)

Reads normal size ISO/IEC 19794-2:2005 card format standard template from memory buffer and converts it to NfRecord

([see page 666](#)).

C#

```
public static NfRecord ReadNormalSizeMinutiae(IntPtr pBuffer, int bufferSize,
FmcMinutiaOrder order, uint flags);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer that contains minutiae.
int bufferSize	Size of memory buffer that contains minutiae.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
uint flags	Reserved. Must be zero.

Returns

The NfRecord ([see page 666](#)) object.

Exceptions

Exceptions	Description
ArgumentOutOfRangeException	bufferSize is less than zero

6.2.3.1.6.2.2.3 FMCARD.ReadNormalSizeMinutiae Method (byte[], FmcMinutiaOrder)

Reads normal size ISO/IEC 19794-2:2005 card format standard template from byte array and converts it to NfRecord ([see page 666](#)).

C#

```
public static NfRecord ReadNormalSizeMinutiae(byte[] buffer, FmcMinutiaOrder order);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains minutiae.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.

Returns

The NfRecord ([see page 666](#)) object.

6.2.3.1.6.2.2.4 FMCARD.ReadNormalSizeMinutiae Method (byte[], FmcMinutiaOrder, uint)

Reads normal size ISO/IEC 19794-2:2005 card format standard template from byte array and converts it to NfRecord ([see page 666](#)).

C#

```
public static NfRecord ReadNormalSizeMinutiae(byte[] buffer, FmcMinutiaOrder order, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains minutiae.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
uint flags	Reserved. Must be zero.

Returns

The NfRecord ([see page 666](#)) object.

6.2.3.1.6.2.2.5 FMCARD.ReadNormalSizeMinutiae Method (byte[], int, int, FmcMinutiaOrder)

Reads normal size ISO/IEC 19794-2:2005 card format standard template from byte array and converts it to NFRecord (see page 666).

C#

```
public static NFRecord ReadNormalSizeMinutiae(byte[] buffer, int index, int count, FmcMinutiaOrder order);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains minutiae.
int index	Start index of byte array.
int count	The length of byte array.
FmcMinutiaOrder order	Order in which minutiae are in the byte array.

Returns

The NFRecord (see page 666) object.

6.2.3.1.6.2.2.6 FMCARD.ReadNormalSizeMinutiae Method (byte[], int, int, FmcMinutiaOrder, uint)

Reads normal size ISO/IEC 19794-2:2005 card format standard template from byte array and converts it to NFRecord (see page 666).

C#

```
public static NFRecord ReadNormalSizeMinutiae(byte[] buffer, int index, int count, FmcMinutiaOrder order, uint flags);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains minutiae.
int index	Start index of byte array.
int count	The length of byte array.
FmcMinutiaOrder order	Order in which minutiae are in the byte array.
uint flags	Reserved. Must be zero.

Returns

The NFRecord (see page 666) object.

6.2.3.1.6.2.3 WriteCompactSizeMinutiae Method

6.2.3.1.6.2.3.1 FMCARD.WriteCompactSizeMinutiae Method (NFRecord, FmcMinutiaOrder)

Converts the NFRecord (see page 666) to compact size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C#

```
public static byte[] WriteCompactSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.

Returns

Byte array with minutiae.

Remarks

If the `FmcMinutiaOrder` (see page 814) is specified in `FMCard.WriteCompactSizeMinutiae`, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using `FmcMinutiaOrder.Polar` example:

```
byte[] fmcCard = FMCard.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmcCard = FMCard.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Descending);
```

Only for `FmcMinutiaOrder.None` and `FmcMinutiaOrder.XORCoordinateExtension` sorting is not needed:

```
byte[] fmcCard = FMCard.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.None);
byte[] fmcCard = FMCard.WriteCompactSizeMinutiae(nfRecord,
FmcMinutiaOrder.XORCoordinateExtension);
```

6.2.3.1.6.2.3.2 FMCard.WriteCompactSizeMinutiae Method (NFRecord, FmcMinutiaOrder, IntPtr, int)

Converts the `NFRecord` (see page 666) to compact size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C#

```
public static int WriteCompactSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, IntPtr
pBuffer, int bufferSize);
```

Parameters

Parameters	Description
<code>NFRecord nfRecord</code>	The <code>NFRecord</code> (see page 666) object.
<code>FmcMinutiaOrder order</code>	Order in which minutiae are in the memory buffer.
<code>IntPtr pBuffer</code>	Pointer to memory buffer to write minutiae in. Can be null.
<code>int bufferSize</code>	Size of memory buffer to write minutiae in.

Returns

Size of written minutiae.

Remarks

If the `FmcMinutiaOrder` (see page 814) is specified in `FMCard.WriteCompactSizeMinutiae`, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using `FmcMinutiaOrder.Polar` example:

```
byte[] fmcCard = FMCard.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmcCard = FMCard.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Descending);
```

Only for `FmcMinutiaOrder.None` and `FmcMinutiaOrder.XORCoordinateExtension` sorting is not needed:

```
byte[] fmcCard = FMCard.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.None);
byte[] fmcCard = FMCard.WriteCompactSizeMinutiae(nfRecord,
FmcMinutiaOrder.XORCoordinateExtension);
```

6.2.3.1.6.2.3.3 FMCard.WriteCompactSizeMinutiae Method (NFRecord, FmcMinutiaOrder, IntPtr, int, uint)

Converts the `NFRecord` (see page 666) to compact size ISO/IEC 19794-2:2005 card format standard template and writes

it to memory buffer.

C#

```
public static int WriteCompactSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, IntPtr pBuffer, int bufferSize, uint flags);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
IntPtr pBuffer	Pointer to memory buffer to write minutiae in. Can be null.
int bufferSize	Size of memory buffer to write minutiae in.
uint flags	Reserved. Must be zero.

Returns

Size of written minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteCompactSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmcCard = FMCARD.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar | FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmcCard = FMCARD.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar | FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
byte[] fmcCard = FMCARD.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.None);
byte[] fmcCard = FMCARD.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.XOrYCoordinateExtension);
```

6.2.3.1.6.2.3.4 FMCARD.WriteCompactSizeMinutiae Method (NFRecord, FmcMinutiaOrder, byte[], int)

Converts the NFRecord (see page 666) to compact size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C#

```
public static int WriteCompactSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, byte[] buffer, int index);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
byte[] buffer	Byte array for writing compact size of minutiae data.
int index	Start index of a byte array. Data will be written starting from this index.

Returns

Size of written minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteCompactSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmc_card = FMCARD.WriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmc_card = FMCARD.WriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
byte[] fmc_card = FMCARD.WriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.None);
byte[] fmc_card = FMCARD.WriteCompactSizeMinutiae(nfrecord,
FmcMinutiaOrder.XOrYCoordinateExtension);
```

6.2.3.1.6.2.3.5 FMCARD.WriteCompactSizeMinutiae Method (NFRecord, FmcMinutiaOrder, byte[], int, uint)

Converts the NFRecord (see page 666) to compact size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C#

```
public static int WriteCompactSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, byte[]
buffer, int index, uint flags);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
byte[] buffer	Byte array for writing compact size of minutiae data.
int index	Start index of a byte array. Data will be written starting from this index.
uint flags	Reserved, must be zero.

Returns

Size of written minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteCompactSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmc_card = FMCARD.WriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmc_card = FMCARD.WriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
byte[] fmc_card = FMCARD.WriteCompactSizeMinutiae(nfrecord, FmcMinutiaOrder.None);
byte[] fmc_card = FMCARD.WriteCompactSizeMinutiae(nfrecord,
FmcMinutiaOrder.XOrYCoordinateExtension);
```

Exceptions

Exceptions	Description
ArgumentOutOfRangeException	index is less than zero
ArgumentException	index is greater than buffer length

6.2.3.1.6.2.3.6 FMCARD.WriteCompactSizeMinutiae Method (NFRecord, FmcMinutiaOrder, uint)

C#

```
public static byte[] WriteCompactSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order,
uint flags);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
uint flags	Reserved, must be zero.

Returns

Byte array with minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteCompactSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmcCard = FMCARD.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmcCard = FMCARD.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XORCoordinateExtension sorting is not needed:

```
byte[] fmcCard = FMCARD.WriteCompactSizeMinutiae(nfRecord, FmcMinutiaOrder.None);
byte[] fmcCard = FMCARD.WriteCompactSizeMinutiae(nfRecord,
FmcMinutiaOrder.XORCoordinateExtension);
```

6.2.3.1.6.2.4 WriteNormalSizeMinutiae Method

6.2.3.1.6.2.4.1 FMCARD.WriteNormalSizeMinutiae Method (NFRecord, FmcMinutiaOrder)

Converts NFRecord (see page 666) to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C#

```
public static byte[] WriteNormalSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.

Returns

Byte array with minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteNormalSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.None);
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord,
FmcMinutiaOrder.XOrYCoordinateExtension);
```

6.2.3.1.6.2.4.2 FMCARD.WriteNormalSizeMinutiae Method (NFRecord, FmcMinutiaOrder, IntPtr, int)

Converts the NFRecord (see page 666) to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C#

```
public static int WriteNormalSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, IntPtr
pBuffer, int bufferSize);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
IntPtr pBuffer	Pointer to memory buffer to write minutiae in. Can be null.
int bufferSize	Size of memory buffer to write minutiae in.

Returns

Size of written minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteNormalSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.None);
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord,
FmcMinutiaOrder.XOrYCoordinateExtension);
```

6.2.3.1.6.2.4.3 FMCARD.WriteNormalSizeMinutiae Method (NFRecord, FmcMinutiaOrder, IntPtr, int, uint)

Converts the NFRecord (see page 666) to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to memory buffer.

C#

```
public static int WriteNormalSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, IntPtr
pBuffer, int bufferSize, uint flags);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.

IntPtr pBuffer	Pointer to memory buffer to write minutiae in. Can be null.
int bufferSize	Size of memory buffer to write minutiae in.
uint flags	Reserved. Must be zero.

Returns

Size of written minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteNormalSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.None);
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord,
FmcMinutiaOrder.XOrYCoordinateExtension);
```

Exceptions

Exceptions	Description
ArgumentNullException	nfRecord is Null.
ArgumentOutOfRangeException	bufferSize is less than zero

6.2.3.1.6.2.4.4 FMCARD.WriteNormalSizeMinutiae Method (NFRecord, FmcMinutiaOrder, byte[], int)

Converts the NFRecord (see page 666) to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to byte array.

C#

```
public static int WriteNormalSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, byte[]
buffer, int index);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
byte[] buffer	Byte array for writing normal size of minutiae data.
int index	Start index of a byte array. Data will be written starting from this index.

Returns

Size of written minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteNormalSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar |
FmcMinutiaOrder.Ascending);
```


or

```
byte[] fmc_card = FMCARD.WriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.Polar |  
FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
byte[] fmc_card = FMCARD.WriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.None);  
byte[] fmc_card = FMCARD.WriteNormalSizeMinutiae(nfrecord,  
FmcMinutiaOrder.XOrYCoordinateExtension);
```

6.2.3.1.6.2.4.5 FMCARD.WriteNormalSizeMinutiae Method (NFRecord, FmcMinutiaOrder, byte[], int, uint)

Converts the NFRecord (see page 666) to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to byte array.

C#

```
public static int WriteNormalSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, byte[]  
buffer, int index, uint flags);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
byte[] buffer	Byte array for writing normal size of minutiae data.
int index	Start index of a byte array. Data will be written starting from this index.
uint flags	Reserved, must be zero.

Returns

Size of written minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteNormalSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmc_card = FMCARD.WriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.Polar |  
FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmc_card = FMCARD.WriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.Polar |  
FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XOrYCoordinateExtension sorting is not needed:

```
byte[] fmc_card = FMCARD.WriteNormalSizeMinutiae(nfrecord, FmcMinutiaOrder.None);  
byte[] fmc_card = FMCARD.WriteNormalSizeMinutiae(nfrecord,  
FmcMinutiaOrder.XOrYCoordinateExtension);
```

Exceptions

Exceptions	Description
ArgumentOutOfRangeException	index is less than zero
ArgumentException	index is greater than buffer length

6.2.3.1.6.2.4.6 FMCARD.WriteNormalSizeMinutiae Method (NFRecord, FmcMinutiaOrder, uint)

Converts the NFRecord (see page 666) to normal size ISO/IEC 19794-2:2005 card format standard template and writes it to byte array.

C#

```
public static byte[] WriteNormalSizeMinutiae(NFRecord nfRecord, FmcMinutiaOrder order, uint flags);
```

Parameters

Parameters	Description
NFRecord nfRecord	The NFRecord (see page 666) object.
FmcMinutiaOrder order	Order in which minutiae are in the memory buffer.
uint flags	Reserved, must be zero.

Returns

Byte array with minutiae.

Remarks

If the FmcMinutiaOrder (see page 814) is specified in FMCARD.WriteNormalSizeMinutiae, it needs to define which sort will be used (ascending or descending).

The minutiae sorting using FmcMinutiaOrder.Polar example:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar | FmcMinutiaOrder.Ascending);
```

or

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.Polar | FmcMinutiaOrder.Descending);
```

Only for FmcMinutiaOrder.None and FmcMinutiaOrder.XORCoordinateExtension sorting is not needed:

```
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.None);
byte[] fmcCard = FMCARD.WriteNormalSizeMinutiae(nfRecord, FmcMinutiaOrder.XORCoordinateExtension);
```

Exceptions

Exceptions	Description
Exception	Written and calculated size of FMCARD (see page 788) normal size minutiae do not match.

6.2.3.1.7 FmcBdtTags Class

Provides constants for ISO/IEC 7816-11:2004 Biometric data template.

Class Hierarchy

C#

```
public static class FmcBdtTags;
```

FmcBdtTags Fields

	Name	Description
	CellQualityData (see page 802)	Tag for storing cell quality data in ISO/IEC 7816-11:2004 Biometric data template.
	CorePointData (see page 802)	Tag for storing core point data in ISO/IEC 7816-11:2004 Biometric data template.
	DeltaPointData (see page 802)	Tag for storing delta point data in ISO/IEC 7816-11:2004 Biometric data template.
	FingerMinutiaeData (see page 802)	Tag for storing finger minutiae data in ISO/IEC 7816-11:2004 Biometric data template.
	RidgeCountData (see page 802)	Tag for storing ridge count data in ISO/IEC 7816-11:2004 Biometric data template.

6.2.3.1.7.1 FmcBdtTags Fields

6.2.3.1.7.1.1 FmcBdtTags.CellQualityData Field

Tag for storing cell quality data in ISO/IEC 7816-11:2004 Biometric data template.

C#

```
public static readonly BerTag CellQualityData = (BerTag)0x94;
```

6.2.3.1.7.1.2 FmcBdtTags.CorePointData Field

Tag for storing core point data in ISO/IEC 7816-11:2004 Biometric data template.

C#

```
public static readonly BerTag CorePointData = (BerTag)0x92;
```

6.2.3.1.7.1.3 FmcBdtTags.DeltaPointData Field

Tag for storing delta point data in ISO/IEC 7816-11:2004 Biometric data template.

C#

```
public static readonly BerTag DeltaPointData = (BerTag)0x93;
```

6.2.3.1.7.1.4 FmcBdtTags.FingerMinutiaeData Field

Tag for storing finger minutiae data in ISO/IEC 7816-11:2004 Biometric data template.

C#

```
public static readonly BerTag FingerMinutiaeData = (BerTag)0x90;
```

6.2.3.1.7.1.5 FmcBdtTags.RidgeCountData Field

Tag for storing ridge count data in ISO/IEC 7816-11:2004 Biometric data template.

C#

```
public static readonly BerTag RidgeCountData = (BerTag)0x91;
```

6.2.3.1.8 FmcBitBhtBaptTags Class

Provides constants for ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.

Class Hierarchy

C#

```
public static class FmcBitBhtBaptTags;
```

FmcBitBhtBaptTags Fields

	Name	Description
	FeatureHandlingIndicator (see page 803)	Tag for storing FmcFeatureHandling (see page 813) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
	MinutiaeOrder (see page 803)	Tag for storing FmcMinutiaOrder (see page 814) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.
	NumberOfMinutiae (see page 803)	Tag for storing minimal and maximal number of minutiae in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.

6.2.3.1.8.1 FmcBitBhtBaptTags Fields

6.2.3.1.8.1.1 FmcBitBhtBaptTags.FeatureHandlingIndicator Field

Tag for storing FmcFeatureHandling (see page 813) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.

C#

```
public static readonly BerTag FeatureHandlingIndicator = (BerTag)0x83;
```

6.2.3.1.8.1.2 FmcBitBhtBaptTags.MinutiaeOrder Field

Tag for storing FmcMinutiaOrder (see page 814) value in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.

C#

```
public static readonly BerTag MinutiaeOrder = (BerTag)0x82;
```

6.2.3.1.8.1.3 FmcBitBhtBaptTags.NumberOfMinutiae Field

Tag for storing minimal and maximal number of minutiae in ISO/IEC 7816-11:2004 Biometric algorithm parameters template in Biometric header template in Biometric information template.

C#

```
public static readonly BerTag NumberOfMinutiae = (BerTag)0x81;
```

6.2.3.1.9 NBiometricStandards Class

Provides methods for retrieving library information.

Class Hierarchy

C#

```
public static class NBiometricStandards;
```

NBiometricStandards Fields

	Name	Description
	DllName (see page 803)	Name of DLL containing unmanaged part of this class.

NBiometricStandards Methods

	Name	Description
	GetInfo (see page 803)	Retrieves information about the library.

6.2.3.1.9.1 NBiometricStandards Fields

6.2.3.1.9.1.1 NBiometricStandards.DllName Field

Name of DLL containing unmanaged part of this class.

C#

```
public const string DllName = "NBiometricStandards";
```

6.2.3.1.9.2 NBiometricStandards Methods

6.2.3.1.9.2.1 NBiometricStandards.GetInfo Method

Retrieves information about the library.

C#

```
public static NLibraryInfo GetInfo();
```

Returns

Returns NLibraryInfo (see page 619) structure with library information.

6.2.3.2 Structs, Records, Enums

6.2.3.2.1 Neurotec.Biometrics.Standards.BdifEyeColor Enumeration

Enumerates eye color constants.

C#

```
public enum BdifEyeColor {  
    Unspecified = 0,  
    Black = 1,  
    Blue = 2,  
    Brown = 3,  
    Gray = 4,  
    Green = 5,  
    Hazel = 6,  
    Maroon = 7,  
    MultiColored = 8,  
    Pink = 9,  
    Unknown = 255  
}
```

Members

Members	Description
Unspecified = 0	Unspecified.
Black = 1	Black.
Blue = 2	Blue.
Brown = 3	Brown.
Gray = 4	Gray.
Green = 5	Green.
Hazel = 6	Hazel.
Maroon = 7	Maroon.
MultiColored = 8	MultiColored.
Pink = 9	Pink.
Unknown = 255	Unknown.

6.2.3.2.2 Neurotec.Biometrics.Standards.BdifEyePosition Enumeration

Enumerates eye position constants.

C#

```
public enum BdifEyePosition {  
    Unknown = 0,  
    Right = 1,  
    Left = 2  
}
```

Members

Members	Description
Unknown = 0	Unknown eye position.
Right = 1	Right eye.

Left = 2	Left eye.

6.2.3.2.3 Neurotec.Biometrics.Standards.BdifFaceExpression Enumeration

Enumerates facial feature constants.

C#

```
public enum BdifFaceExpression {
    Unspecified = 0,
    Neutral = 1,
    Smile = 2,
    SmileOpenedJaw = 3,
    RaisedBrows = 4,
    EyesAway = 5,
    Squinting = 6,
    Frowning = 7
}
```

Members

Members	Description
Unspecified = 0	Unspecified.
Neutral = 1	Neutral.
Smile = 2	Smile.
SmileOpenedJaw = 3	Smile with jaw open.
RaisedBrows = 4	Eyebrows rised.
EyesAway = 5	Eyes looking away.
Squinting = 6	Squinting.
Frowning = 7	Frowning.

Remarks

See BdifFaceExpression in Reference (C/C++) chapter.

6.2.3.2.4 Neurotec.Biometrics.Standards.BdifFaceFeaturePointType Enumeration

Enumerates face feature point type constants.

C#

```
public enum BdifFaceFeaturePointType {
    Point2D = 1
}
```

Members

Members	Description
Point2D = 1	2D point.

Remarks

See BdifFaceFeaturePointType in Reference (C/C++) chapter.

6.2.3.2.5 Neurotec.Biometrics.Standards.BdifFaceProperties Enumeration

Enumerates basic face properties constants

C#

```
[Flags]
```

```

public enum BdifFaceProperties {
    NotSpecified = 0,
    Specified = 0x0000001,
    Glasses = 0x0000002,
    Moustache = 0x0000004,
    Beard = 0x0000008,
    TeethVisible = 0x0000010,
    Blink = 0x0000020,
    MouthOpen = 0x0000040,
    LeftEyePatch = 0x0000080,
    RightEyePatch = 0x0000100,
    BothEyePatch = 0x0000200,
    DarkGlasses = 0x0000400,
    DistortingCondition = 0x0000800,
    Hat = 0x01000000,
    Scarf = 0x02000000,
    NoEar = 0x04000000
}

```

Members

Members	Description
NotSpecified = 0	Face properties not specified.
Specified = 0x0000001	Face properties specified.
Glasses = 0x0000002	Wearing glasses.
Moustache = 0x0000004	Moustaches.
Beard = 0x0000008	Beard.
TeethVisible = 0x0000010	Teeth visible.
Blink = 0x0000020	Eye blink.
MouthOpen = 0x0000040	Mouth is open.
LeftEyePatch = 0x0000080	Left eye patch.
RightEyePatch = 0x0000100	bfpRightEyePatch
BothEyePatch = 0x0000200	Both eyes patch.
DarkGlasses = 0x0000400	Wearing dark glasses.
DistortingCondition = 0x0000800	Distorting conditons.
Hat = 0x01000000	Wearing a hat.
Scarf = 0x02000000	Wearing a scarf.
NoEar = 0x04000000	No ear.

Remarks

See BdifFaceProperties in Reference (C/C++) chapter.

6.2.3.2.6 Neurotec.Biometrics.Standards.BdifFPatternClass Enumeration

Enumerates fingerprint pattern class constants.

C#

```

public enum BdifFPatternClass {
    Unknown = 0,
    PlainArch = 1,
    TentedArch = 2,
    RadialLoop = 3,
    UlnarLoop = 4,
    PlainWhorl = 5,
    CentralPocketLoop = 6,
    DoubleLoop = 7,
    AccidentalWhorl = 8,
    Whorl = 9,
    RightSlantLoop = 10,
    LeftSlantLoop = 11,
}

```

```

    Scar = 12,
    Amputation = 15
}

```

Members

Members	Description
Unknown = 0	Unknown.
PlainArch = 1	Plain arch.
TentedArch = 2	Tented arch.
RadialLoop = 3	Radial loop.
UlnarLoop = 4	Ulnar loop.
PlainWhorl = 5	Plain whorl.
CentralPocketLoop = 6	Central pocket loop.
DoubleLoop = 7	Double loop.
AccidentalWhorl = 8	Accidental Whorl.
Whorl = 9	Whorl.
RightSlantLoop = 10	Right Slant Loop.
LeftSlantLoop = 11	Left slant loop.
Scar = 12	Scar.
Amputation = 15	Amputation.

Remarks

See BdiffPatternClass in Reference (C/C++) chapter.

6.2.3.2.7 Neurotec.Biometrics.Standards.BdiffFPImpressionType Enumeration

Enumerates fingerprint impression type constants.

C#

```

public enum BdiffFPImpressionType {
    LiveScanPlain = 0,
    LiveScanRolled = 1,
    NonliveScanPlain = 2,
    NonliveScanRolled = 3,
    LatentImpression = 4,
    LatentTracing = 5,
    LatentPhoto = 6,
    LatentLift = 7,
    LiveScanVerticalSwipe = 8,
    LiveScanContactless = 9,
    LiveScanPalm = 10,
    NonliveScanPalm = 11,
    LatentPalmImpression = 12,
    LatentPalmTracing = 13,
    LatentPalmPhoto = 14,
    LatentPalmLift = 15,
    LiveScanOpticalContactPlain = 20,
    LiveScanOpticalContactRolled = 21,
    LiveScanNonOpticalContactPlain = 22,
    LiveScanNonOpticalContactRolled = 23,
    LiveScanOpticalContactlessPlain = 24,
    LiveScanOpticalContactlessRolled = 25,
    LiveScanNonOpticalContactlessPlain = 26,
    LiveScanNonOpticalContactlessRolled = 27,
    Other = 28,
    Unknown = 29
}

```


Members

Members	Description
LiveScanPlain = 0	Plain live scan.
LiveScanRolled = 1	Rolled live scan.
NonliveScanPlain = 2	Plain non live scan.
NonliveScanRolled = 3	Rolled non live scan.
LatentImpression = 4	Latent impression.
LatentTracing = 5	Latent tracing.
LatentPhoto = 6	Latent photo.
LatentLift = 7	Latent lift.
LiveScanVerticalSwipe = 8	Vertical swipe live scan.
LiveScanContactless = 9	Contactless live scan.
LiveScanPalm = 10	Palm live scan.
NonliveScanPalm = 11	Palm non live scan.
LatentPalmImpression = 12	Latent palm impression.
LatentPalmTracing = 13	Latent palm tracing.
LatentPalmPhoto = 14	Latent palm photo.
LatentPalmLift = 15	Latent palm lift.
LiveScanOpticalContactPlain = 20	Optical contact plain live scan.
LiveScanOpticalContactRolled = 21	Optical contact rolled live scan.
LiveScanNonOpticalContactPlain = 22	Non optical contact plain live scan.
LiveScanNonOpticalContactRolled = 23	Non optical contact rolled live scan.
LiveScanOpticalContactlessPlain = 24	Optical contactless plain live scan.
LiveScanOpticalContactlessRolled = 25	Optical contactless rolled live scan.
LiveScanNonOpticalContactlessPlain = 26	Non optical contactless plain live scan.
LiveScanNonOpticalContactlessRolled = 27	Non optical contactless rolled live scan.
Other = 28	Other.
Unknown = 29	Unknown.

Remarks

See BdifFPImpressionType in Reference (C/C++) chapter.

6.2.3.2.8 Neurotec.Biometrics.Standards.BdifFPMinutiaType Enumeration

Enumerates fingerprint minutia type constants.

C#

```
public enum BdifFPMinutiaType {
    Unspecified = -1,
    Unknown = 0,
    End = 1,
    Bifurcation = 2,
    Other = 3
}
```

Members

Members	Description
Unspecified = -1	Unspecified minutia type.
Unknown = 0	Unknown minutia type.
End = 1	Ridge end minutia type.
Bifurcation = 2	Ridge bifurcation minutia type.

Other = 3	Other minutia type.

Remarks

See BdifFPMinutiaType in Reference (C/C++) chapter.

6.2.3.2.9 Neurotec.Biometrics.Standards.BdifFPPosition Enumeration

Enumerates fingerprint position constants.

C#

```
public enum BdifFPPosition {
    Unknown = 0,
    RightThumb = 1,
    RightIndexFinger = 2,
    RightMiddleFinger = 3,
    RightRingFinger = 4,
    RightLittleFinger = 5,
    LeftThumb = 6,
    LeftIndexFinger = 7,
    LeftMiddleFinger = 8,
    LeftRingFinger = 9,
    LeftLittleFinger = 10,
    PlainRightThumb = 11,
    PlainLeftThumb = 12,
    PlainRightFourFingers = 13,
    PlainLeftFourFingers = 14,
    PlainThumbs = 15,
    MajorCase = 19,
    UnknownPalm = 20,
    RightFullPalm = 21,
    RightWritersPalm = 22,
    LeftFullPalm = 23,
    LeftWritersPalm = 24,
    RightLowerPalm = 25,
    RightUpperPalm = 26,
    LeftLowerPalm = 27,
    LeftUpperPalm = 28,
    RightOther = 29,
    LeftOther = 30,
    RightInterdigital = 31,
    RightThenar = 32,
    RightHypothenar = 33,
    LeftInterdigital = 34,
    LeftThenar = 35,
    LeftHypothenar = 36
}
```

Members

Members	Description
Unknown = 0	Unknown.
RightThumb = 1	Right thumb.
RightIndexFinger = 2	Right index finger.
RightMiddleFinger = 3	Right middle finger.
RightRingFinger = 4	Right ring finger.
RightLittleFinger = 5	Right little finger.
LeftThumb = 6	Left thumb.
LeftIndexFinger = 7	Left index finger.
LeftMiddleFinger = 8	Left middle finger.
LeftRingFinger = 9	Left ring finger.
LeftLittleFinger = 10	Left little finger.

PlainRightThumb = 11	Plain right thumb.
PlainLeftThumb = 12	Plain left thumb.
PlainRightFourFingers = 13	Plain right four fingers.
PlainLeftFourFingers = 14	Plain left four fingers.
PlainThumbs = 15	Plain thumbs.
MajorCase = 19	Major case.
UnknownPalm = 20	Unknown palm.
RightFullPalm = 21	Right full palm.
RightWritersPalm = 22	Right writers palm.
LeftFullPalm = 23	Left full palm.
LeftWritersPalm = 24	Left writers palm.
RightLowerPalm = 25	Right lower palm.
RightUpperPalm = 26	Right upper palm.
LeftLowerPalm = 27	Left lower palm.
LeftUpperPalm = 28	Left upper palm.
RightOther = 29	Right other.
LeftOther = 30	Left other.
RightInterdigital = 31	Right interdigital.
RightThenar = 32	Right thenar.
RightHypothenar = 33	Right hypothenar.
LeftInterdigital = 34	Left interdigital,
LeftThenar = 35	Left thenar.
LeftHypothenar = 36	Left hypothenar.

Remarks

See BdifFPPosition in Reference (C/C++) chapter.

6.2.3.2.10 Neurotec.Biometrics.Standards.BdifGender Enumeration

Enumerates gender constants.

C#

```
public enum BdifGender {
    Unspecified = 0,
    Male = 1,
    Female = 2,
    Unknown = 255
}
```

Members

Members	Description
Unspecified = 0	Unspecified gender.
Male = 1	Male.
Female = 2	Female.
Unknown = 255	Unknown.

Remarks

See BdifGender in Reference (C/C++) chapter.

6.2.3.2.11 Neurotec.Biometrics.Standards.BdifHairColor Enumeration

Enumerates hair color constants.

```
C#
public enum BdifHairColor {
    Unspecified = 0,
    Bald = 1,
    Black = 2,
    Blonde = 3,
    Brown = 4,
    Gray = 5,
    Red = 6,
    Blue = 7,
    Green = 8,
    Orange = 9,
    Pink = 10,
    Purple = 11,
    Sandy = 12,
    Auburn = 13,
    White = 14,
    Strawberry = 15,
    Unknown = 255
}
```

Members

Members	Description
Unspecified = 0	Unspecified.
Bald = 1	Bald.
Black = 2	Black.
Blonde = 3	Blonde.
Brown = 4	Brown.
Gray = 5	Gray.
Red = 6	Red.
Blue = 7	Blue.
Green = 8	Green.
Orange = 9	Orange.
Pink = 10	Pink.
Purple = 11	Purple.
Sandy = 12	Sandy.
Auburn = 13	Auburn.
White = 14	White.
Strawberry = 15	Strawberry.
Unknown = 255	Unknown.

Remarks

See BdifHairColor in Reference (C/C++) chapter.

6.2.3.2.12 Neurotec.Biometrics.Standards.BdifImageSourceType Enumeration

Enumerates image source type constants

```
C#
public enum BdifImageSourceType {
```

```
Unspecified = 0,  
UnknownPhoto = 1,  
DigitalPhoto = 2,  
ScannedPhoto = 3,  
UnknownVideo = 4,  
AnalogueVideo = 5,  
DigitalVideo = 6,  
Unknown = 7,  
Vendor = 128  
}
```

Members

Members	Description
Unspecified = 0	Unspecified.
UnknownPhoto = 1	Unknown photo.
DigitalPhoto = 2	Digital photo.
ScannedPhoto = 3	Scanned photo.
UnknownVideo = 4	Unknown video.
AnalogueVideo = 5	Analogue video.
DigitalVideo = 6	Digital video.
Unknown = 7	Unknown.
Vendor = 128	Vendor.

Remarks

See BdifImageSourceType in Reference (C/C++) chapter.

6.2.3.2.13 Neurotec.Biometrics.Standards.BdifIrisOrientation Enumeration

Enumerates Bdif (see page 754) Iris orientation values.

C#

```
public enum BdifIrisOrientation {  
    Undefined = 0,  
    Base = 1,  
    Flipped = 2  
}
```

Members

Members	Description
Undefined = 0	Undefined iris orientation.
Base = 1	Base
Flipped = 2	Flipped

6.2.3.2.14 Neurotec.Biometrics.Standards.BdifIrisScanType Enumeration

Enumerates Bdif (see page 754) scan type values.

C#

```
public enum BdifIrisScanType {  
    Undefined = 0,  
    Progressive = 1,  
    InterlaceFrame = 2,  
    InterlaceField = 3,  
    Corrected = 4  
}
```

Members

Members	Description
Undefined = 0	Scan type undefined.
Progressive = 1	Progressive scan type.
InterlaceFrame = 2	InterlaceFrame scan type.
InterlaceField = 3	InterlaceFiled scan type.
Corrected = 4	Corrected scan type.

6.2.3.2.15 Neurotec.Biometrics.Standards.BdifScaleUnits Enumeration

Enumerates scale unit constants.

C#

```
public enum BdifScaleUnits {  
    None = 0,  
    PixelsPerInch = 1,  
    PixelsPerCentimeter = 2  
}
```

Members

Members	Description
None = 0	Default scale units.
PixelsPerInch = 1	Pixels per pinch.
PixelsPerCentimeter = 2	Pixels per centimeter.

Remarks

See BdifScaleUnits in Reference (C/C++) chapter.

6.2.3.2.16 Neurotec.Biometrics.Standards.BdifStandard Enumeration

Enumerates standard constants.

C#

```
public enum BdifStandard {  
    Iso = 0,  
    Ansi = 1  
}
```

Members

Members	Description
Iso = 0	Iso standard.
Ansi = 1	Ansi standard.

Remarks

See BdifStandard in Reference (C/C++) chapter.

6.2.3.2.17 Neurotec.Biometrics.Standards.FmcFeatureHandling Enumeration

Specifies additional fingerprint features that are supported by the card.

C#

```
[Flags]  
public enum FmcFeatureHandling {
```

```

None = 0,
RidgeCounts = 0x01,
CorePoints = 0x02,
DeltaPoints = 0x04,
CellQuality = 0x08
}

```

Members

Members	Description
None = 0	None additional features are supported.
RidgeCounts = 0x01	Ridge counts supported.
CorePoints = 0x02	Core points supported.
DeltaPoints = 0x04	Delta points supported.
CellQuality = 0x08	Cell quality supported.

6.2.3.2.18 Neurotec.Biometrics.Standards.FmcMinutiaOrder Enumeration

Specifies minutia ordering required by the card.

C#

```

[Flags]
public enum FmcMinutiaOrder {
    None = 0,
    Ascending = 0x01,
    Descending = 0x02,
    CartesianXY = 0x04,
    CartesianYX = 0x08,
    Angle = 0x0C,
    Polar = 0x10,
    XOrYCoordinateExtension = 0x20
}

```

Members

Members	Description
None = 0	Specifies than no ordering is required.
Ascending = 0x01	Specifies than minutiae are sorted ascending by the specified order.
Descending = 0x02	Specifies than minutiae are sorted descending by the specified order.
CartesianXY = 0x04	Specifies than minutiae are sorted by X coordinate. If X coordinates of two minutiae are equal Y coordinate is compared.
CartesianYX = 0x08	Specifies than minutiae are sorted by Y coordinate. If Y coordinates of two minutiae are equal X coordinate is compared.
Angle = 0x0C	Specifies than minutiae are sorted by minutia angle.
Polar = 0x10	Specifies than minutiae are sorted by distance from minutiae center of mass. If distance of two minutiae are equal minutia angle is compared.
XOrYCoordinateExtension = 0x20	Specifies than minutiae X or Y coordinate extension is used for compact format.

6.2.4 Neurotec.DeviceManager Namespace

Provides functionality for managing devices, like cameras or fingerprint scanners.

Classes

	Name	Description
	FPScanner (see page 815)	One instance represents one physical device.
	FPScannerImageScannedEventArgs (see page 818)	The class contains data of event ImageScanned.
	FPScannerMan (see page 819)	Scanners (see page 821) manager enumerates, creates scanners.
	FPScannerManScannerEventArgs (see page 822)	Provides data for ImageScanned event.
	NDeviceManager (see page 822)	Provides functionality for managing devices.

Module

.NET Reference ([see page 606](#))

Types

Name	Description
FPScannerImageScannedEventHandler (see page 823)	Represents the method that will handle the ImageScanned event of a FPScanner (see page 815).
FPScannerManScannerEventHandler (see page 824)	Represent the method that will handle ScannerAdded, ScannerRemoved events of a FPScannerMan (see page 819).

6.2.4.1 Classes

6.2.4.1.1 FPScanner Class

One instance represents one physical device.

Class Hierarchy**C#**

```
public sealed class FPScanner : NObject;
```

FPScanner Events**FPScanner Class**

	Name	Description
	FingerPlaced (see page 817)	Occurs when finger is placed on scanner.
	FingerRemoved (see page 817)	Occurs when finger is removed from scanner.
	ImageScanned (see page 818)	Occurs when image is scanned from scanner.
	IsCapturingChanged (see page 818)	Occurs when capturing from scanner was started, stopped for or capturing was started for one image.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.

	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

FPScanner Class

	Name	Description
	StartCapturing (see page 816)	Starts capturing fingerprint image from certain device.
	StartCapturingForOneImage (see page 816)	Starts capturing one fingerprint image from certain device.
	StopCapturing (see page 816)	Stops capturing fingerprint image.
	ToString (see page 817)	Represents associated device identifier.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

FPScanner Class

	Name	Description
	Id (see page 817)	Gets associated device identifier.
	IsCapturing (see page 817)	Checks scanner status.
	Owner (see page 817)	Gets owner of the object.

6.2.4.1.1.1 FPScanner Methods**6.2.4.1.1.1.1 FPScanner.StartCapturing Method**

Starts capturing fingerprint image from certain device.

C#

```
public void StartCapturing();
```

6.2.4.1.1.1.2 FPScanner.StartCapturingForOneImage Method

Starts capturing one fingerprint image from certain device.

C#

```
public void StartCapturingForOneImage();
```

Remarks

Scanner is stopped automatically when fingerprint is scanned.

6.2.4.1.1.1.3 FPScanner.StopCapturing Method

Stops capturing fingerprint image.

C#

```
public bool StopCapturing();
```

Returns

True if capturing stopped successfully; false otherwise.

6.2.4.1.1.1.4 FPScanner.ToString Method

Represents associated device identifier.

C#

```
public override string ToString();
```

Returns

Associated device identifier.

6.2.4.1.1.2 FPScanner Properties

6.2.4.1.1.2.1 FPScanner.Id Property

Gets associated device identifier.

C#

```
public string Id;
```

Property value

An associated device identifier.

6.2.4.1.1.2.2 FPScanner.IsCapturing Property

Checks scanner status.

C#

```
public bool IsCapturing;
```

Property value

true if capturing from scanner was started; otherwise, false.

6.2.4.1.1.2.3 FPScanner.Owner Property

Gets owner of the object.

C#

```
public new FPScannerMan Owner;
```

Property value

Object (owner) of FPScannerMan (see page 819) type.

6.2.4.1.1.3 FPScanner Events

6.2.4.1.1.3.1 FPScanner.FingerPlaced Event

Occurs when finger is placed on scanner.

C#

```
public event EventHandler FingerPlaced;
```

6.2.4.1.1.3.2 FPScanner.FingerRemoved Event

Occurs when finger is removed from scanner.

C#

```
public event EventHandler FingerRemoved;
```

6.2.4.1.1.3.3 FPScanner.ImageScanned Event

Occurs when image is scanned from scanner.

C#

```
public event FPScannerImageScannedEventHandler ImageScanned;
```

6.2.4.1.1.3.4 FPScanner.IsCapturingChanged Event

Occurs when capturing from scanner was started, stopped or capturing was started for one image.

C#

```
public event EventHandler IsCapturingChanged;
```

6.2.4.1.2 FPScannerImageScannedEventArgs Class

The class contains data of event ImageScanned.

Class Hierarchy

C#

```
[Serializable]  
public class FPScannerImageScannedEventArgs : EventArgs;
```

Methods

	Name	Description
	FPScannerImageScannedEventArgs (see page 818)	Initializes a new instance of the FPScannerImageScannedEventArgs class.

FPScannerImageScannedEventArgs Properties

	Name	Description
	Image (see page 818)	Gets scanned image.

6.2.4.1.2.1 FPScannerImageScannedEventArgs.FPScannerImageScannedEventArgs Constructor

Initializes a new instance of the FPScannerImageScannedEventArgs class.

C#

```
public FPScannerImageScannedEventArgs(NGrayscaleImage image);
```

Parameters

Parameters	Description
NGrayscaleImage image	The NGrayscaleImage (see page 832) image.

6.2.4.1.2.2 FPScannerImageScannedEventArgs Properties

6.2.4.1.2.2.1 FPScannerImageScannedEventArgs.Image Property

Gets scanned image.

C#

```
public NGrayscaleImage Image;
```

Property value

The NGrayscaleImage (see page 832) object.

6.2.4.1.3 FPScannerMan Class

Scanners ([see page 821](#)) manager enumerates, creates scanners.

Class Hierarchy

C#

```
public sealed class FPScannerMan : NObject;
```

FPScannerMan Classes

FPScannerMan Class

	Name	Description
	FPScannerCollection (see page 820)	Represents the collection of FPScanner (see page 815).

FPScannerMan Events

FPScannerMan Class

	Name	Description
	ScannerAdded (see page 821)	Occurs when scanner is connected.
	ScannerRemoved (see page 822)	Occurs when scanner is disconnected.

Methods

FPScannerMan Class

	Name	Description
	FPScannerMan (see page 820)	Initializes a new instance of the FPScannerMan class.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

FPScannerMan Class

	Name	Description
	GetAvailableModules (see page 821)	Gets the list of scanners separated by semicolons.

NObject Properties

NObject Class

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

FPScannerMan Class

	Name	Description
	Scanners (see page 821)	FPScanner (see page 815) collection.

6.2.4.1.3.1 FPScannerMan Constructor**6.2.4.1.3.1.1 FPScannerMan.FPScannerMan Constructor (ISynchronizeInvoke)**

Initializes a new instance of the FPScannerMan class.

C#

```
public FPScannerMan(ISynchronizeInvoke synInvoke);
```

Parameters

Parameters	Description
ISynchronizeInvoke synInvoke	An ISynchronizeInvoke object.

6.2.4.1.3.1.2 FPScannerMan.FPScannerMan Constructor (string, ISynchronizeInvoke)

Initializes a new instance of the FPScannerMan class.

C#

```
public FPScannerMan(string modules, ISynchronizeInvoke synInvoke);
```

Parameters

Parameters	Description
string modules	A string that contains the list of scanners separated by semicolons.
ISynchronizeInvoke synInvoke	An ISynchronizeInvoke object.

6.2.4.1.3.2 FPScannerMan Classes**6.2.4.1.3.2.1 FPScannerMan.FPScannerCollection Class**

Represents the collection of FPScanner ([see page 815](#)).

Class Hierarchy

C#

```
public sealed class FPScannerCollection : NObjectStaticReadOnlyCollection<FPScanner>;
```

FPScannerCollection Methods

	Name	Description
	IndexOf (see page 820)	Reports the index of the first occurrence of FPScanner (see page 815).
	this (see page 821)	Gets FPScanner (see page 815) from collection by scanner Id.

6.2.4.1.3.2.1.1 FPScannerCollection Methods**6.2.4.1.3.2.1.1.1 FPScannerMan.FPScannerCollection.IndexOf Method**

Reports the index of the first occurrence of FPScanner ([see page 815](#)).

C#

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

Parameters

Parameters	Description
string id	An associated device identifier.

Returns

The index of the FPScanner ([see page 815](#)).

6.2.4.1.3.2.1.1.2 FPScannerMan.FPScannerCollection.this Indexer

Gets FPScanner ([see page 815](#)) from collection by scanner Id.

C#

```
public FPScanner this[string id];
```

Parameters

Parameters	Description
string id	An associated device identifier.

Returns

A FPScanner ([see page 815](#)) object.

6.2.4.1.3.3 FPScannerMan Methods**6.2.4.1.3.3.1 FPScannerMan.GetAvailableModules Method**

Gets the list of scanners separated by semicolons.

C#

```
public static string GetAvailableModules();
```

Returns

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

6.2.4.1.3.4 FPScannerMan Properties**6.2.4.1.3.4.1 FPScannerMan.Scanners Property**

FPScanner ([see page 815](#)) collection.

C#

```
public FPScannerCollection Scanners;
```

Returns

FPScanner ([see page 815](#)) collection.

6.2.4.1.3.5 FPScannerMan Events**6.2.4.1.3.5.1 FPScannerMan.ScannerAdded Event**

Occurs when scanner is connected.

C#

```
public event FPScannerManScannerEventHandler ScannerAdded;
```

6.2.4.1.3.5.2 FPScannerMan.ScannerRemoved Event

Occurs when scanner is disconnected.

```
C#
public event FPScannerManScannerEventHandler ScannerRemoved;
```

6.2.4.1.4 FPScannerManScannerEventArgs Class

Provides data for ImageScanned event.

Class Hierarchy

```
C#
[Serializable]
public class FPScannerManScannerEventArgs : EventArgs;
```

Methods

	Name	Description
	FPScannerManScannerEventArgs (see page 822)	Initializes a new instance of the FPScannerManScannerEventArgs class.

FPScannerManScannerEventArgs Properties

	Name	Description
	Scanner (see page 822)	Gets the FPScanner (see page 815) object.

6.2.4.1.4.1 FPScannerManScannerEventArgs.FPScannerManScannerEventArgs Constructor

Initializes a new instance of the FPScannerManScannerEventArgs class.

```
C#
public FPScannerManScannerEventArgs(FPScanner scanner);
```

Parameters

Parameters	Description
FPScanner scanner	The FPScanner (see page 815) object.

6.2.4.1.4.2 FPScannerManScannerEventArgs Properties

6.2.4.1.4.2.1 FPScannerManScannerEventArgs.Scanner Property

Gets the FPScanner (see page 815) object.

```
C#
public FPScanner Scanner;
```

Property value

The FPScanner (see page 815) object.

6.2.4.1.5 NDeviceManager Class

Provides functionality for managing devices.

Class Hierarchy

C#

```
public static class NDeviceManager;
```

NDeviceManager Fields

	Name	Description
	DllName ( see page 823)	CameraMan library's Dll name.

NDeviceManager Methods

	Name	Description
	GetInfo ( see page 823)	Gets information about the library.

6.2.4.1.5.1 NDeviceManager Fields

6.2.4.1.5.1.1 NDeviceManager.DllName Field

CameraMan library's Dll name.

C#

```
public const string DllName = "NDeviceManager";
```

6.2.4.1.5.2 NDeviceManager Methods

6.2.4.1.5.2.1 NDeviceManager.GetInfo Method

Gets information about the library.

C#

```
public static NLibraryInfo GetInfo();
```

Returns

NLibraryInfo ( see page 619) structure with library information.

6.2.4.2 Types

6.2.4.2.1 Neurotec.DeviceManager.FPScannerImageScannedEventHandler Type

Represents the method that will handle the ImageScanned event of a FPScanner ( see page 815).

C#

```
public delegate void FPScannerImageScannedEventHandler(object sender,  
    FPScannerImageScannedEventArgs ea);
```

Parameters

Parameters	Description
sender	The source of the event.
ea	A FPScannerImageScannedEventArgs ( see page 818) that contains the event data.

6.2.4.2.2 Neurotec.DeviceManager.FPScannerManScannerEventHandler Type

Represent the method that will handle ScannerAdded, ScannerRemoved events of a FPScannerMan ([see page 819](#)).

C#

```
public delegate void FPScannerManScannerEventHandler(object sender,
    FPScannerManScannerEventArgs e);
```

Parameters

Parameters	Description
sender	The source of the event.
e	A FPScannerManScannerEventArgs (see page 822) that contains the event data.

6.2.5 Neurotec.Images Namespace

Classes under this namespace provides functionality that enable loading, saving and converting images in various formats.

Classes

	Name	Description
	Bmp (see page 825)	Provides functionality for loading and saving images in BMP format.
	Jpeg (see page 828)	Provides functionality for loading and saving images in JPEG format.
	LosslessJpeg (see page 831)	Provides functionality for saving images in Lossless JPEG format.
	NGrayscaleImage (see page 832)	Provides functionality for managing 8-bit grayscale images.
	NImage (see page 834)	Provides functionality for managing images.
	NImageFile (see page 857)	Provides functionality for working with image files.
	NImageFormat (see page 859)	Provides functionality for loading and saving images in format-neutral style.
	NImages (see page 867)	Provides library registration and other additional functionality.
	NMonochromeImage (see page 869)	Provides functionality for managing 1-bit monochrome images.
	NPixelFormat (see page 870)	Provides functionality for working with pixel format.
	NRgb (see page 876)	Represents an RGB color.
	NRgbImage (see page 878)	Provides functionality for managing 24-bit RGB images.
	Png (see page 880)	Provides functionality for loading and saving images in Png format.
	Tiff (see page 883)	Provides functionality for loading and saving images in TIFF format.

Module

.NET Reference ([see page 606](#))

Structs, Records, Enums

	Name	Description
	BmpRecording (see page 884)	Enumerates different bitmap image pixels arrangement types.
	BmpSaveMode (see page 884)	Enumerates different bitmap image data storage types.
	NImageRotateFlipType (see page 885)	Enumerates NImage (see page 834) rotation and flip values.

6.2.5.1 Classes

6.2.5.1.1 Bmp Class

Provides functionality for loading and saving images in BMP format.

Class Hierarchy

C#

```
public static class Bmp;
```

Bmp Methods

	Name	Description
	LoadImage (see page 825)	Creates a new instance of the NImage (see page 834) class from memory buffer.
	LoadImageFromBitmap (see page 826)	Creates a new instance of the NImage (see page 834) class from Bitmap.
	LoadImageFromHBitmap (see page 827)	Creates a new instance of the NImage (see page 834) class from Windows HBITMAP.
	SaveImage (see page 827)	Saves NImage (see page 834) to NBuffer (see page 886).
	SaveImageToBitmap (see page 828)	Saves NImage (see page 834) to Bitmap.
	SaveImageToHBitmap (see page 828)	Saves NImage (see page 834) to Windows HBITMAP.

6.2.5.1.1.1 Bmp Methods

6.2.5.1.1.1.1 LoadImage Method

6.2.5.1.1.1.1.1 Bmp.LoadImage Method (IntPtr, int)

Creates a new instance of the NImage ([see page 834](#)) class from memory buffer.

C#

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

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer.
int bufferLength	Size of memory buffer.

Returns

A NImage ([see page 834](#)) object.

6.2.5.1.1.1.1.2 Bmp.LoadImage Method (Stream)

Creates a new instance of the NImage ([see page 834](#)) class from stream.

C#

```
public static NImage LoadImage(Stream stream);
```

Parameters

Parameters	Description
Stream stream	Memory stream.

Returns

A NImage (see page 834) object.

6.2.5.1.1.1.1.3 Bmp.LoadImage Method (byte[])

Creates a new instance of the NImage (see page 834) class from byte array.

C#

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

Parameters

Parameters	Description
byte[] buffer	A byte array.

Returns

A NImage (see page 834) object.

6.2.5.1.1.1.1.4 Bmp.LoadImage Method (string)

Creates a new instance of the NImage (see page 834) class from file.

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.

Returns

A NImage (see page 834) object.

6.2.5.1.1.1.2 LoadImageFromBitmap Method**6.2.5.1.1.1.2.1 Bmp.LoadImageFromBitmap Method (Bitmap)**

Creates a new instance of the NImage (see page 834) class from Bitmap.

C#

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

Parameters

Parameters	Description
Bitmap bitmap	A Bitmap class object.

Returns

A NImage (see page 834) object.

6.2.5.1.1.1.2.2 Bmp.LoadImageFromBitmap Method (Bitmap, float, float)

Creates a new instance of the NImage (see page 834) class from Bitmap with specified resolution.

C#

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

Parameters

Parameters	Description
Bitmap bitmap	A Bitmap class object.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.

Returns

A NImage (see page 834) object.

6.2.5.1.1.1.3 Bmp.LoadImageFromHBitmap Method

Creates a new instance of the NImage (see page 834) class from Windows HBITMAP.

C#

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

Parameters

Parameters	Description
IntPtr hBitmap	Pointer to handle that specifies Windows HBITMAP.

Returns

A NImage (see page 834) object.

6.2.5.1.1.1.4 SaveImage Method**6.2.5.1.1.1.4.1 Bmp.SaveImage Method (NImage)**

Saves NImage (see page 834) to NBuffer (see page 886).

C#

```
public static NBuffer SaveImage(NImage image);
```

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.

Returns

NBuffer (see page 886) object.

6.2.5.1.1.1.4.2 Bmp.SaveImage Method (NImage, Stream)

Saves NImage (see page 834) to stream.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
Stream stream	The data stream used to save the image.

6.2.5.1.1.1.4.3 Bmp.SaveImage Method (NImage, string)

Saves NImage (see page 834) to file.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
string fileName	A string that contains the name of the file.

6.2.5.1.1.1.5 Bmp.SaveImageToBitmap Method

Saves NImage (see page 834) to Bitmap.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.

Returns

A Bitmap object.

6.2.5.1.1.1.6 Bmp.SaveImageToHBitmap Method

Saves NImage (see page 834) to Windows HBITMAP.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.

Returns

A pointer to handle of Windows HBITMAP.

6.2.5.1.2 Jpeg Class

Provides functionality for loading and saving images in JPEG format.

Class Hierarchy

C#

```
public static class Jpeg;
```

Jpeg Fields

	Name	Description
	DefaultQuality (see page 829)	Specifies default JPEG quality.

Jpeg Methods

	Name	Description
	LoadImage (see page 829)	Creates NImage (see page 834) object from memory buffer.

	SaveImage (see page 830)	Saves NImage (see page 834) to NBuffer (see page 886).

6.2.5.1.2.1 Jpeg Fields

6.2.5.1.2.1.1 Jpeg.DefaultQuality Field

Specifies default JPEG quality.

C#

```
public const int DefaultQuality = 75;
```

6.2.5.1.2.2 Jpeg Methods

6.2.5.1.2.2.1 LoadImage Method

6.2.5.1.2.2.1.1 Jpeg.LoadImage Method (IntPtr, int)

Creates NImage ([see page 834](#)) object from memory buffer.

C#

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

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer.
int bufferLength	Size of memory buffer.

Returns

A NImage ([see page 834](#)) object.

6.2.5.1.2.2.1.2 Jpeg.LoadImage Method (Stream)

Creates a new instance of the NImage ([see page 834](#)) class from stream.

C#

```
public static NImage LoadImage(Stream stream);
```

Parameters

Parameters	Description
Stream stream	Memory stream.

Returns

A NImage ([see page 834](#)) object.

6.2.5.1.2.2.1.3 Jpeg.LoadImage Method (byte[])

Creates a new instance of the NImage ([see page 834](#)) class from byte array.

C#

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

Parameters

Parameters	Description
byte[] buffer	A byte array.

Returns

A NImage (see page 834) object.

6.2.5.1.2.2.1.4 Jpeg.LoadImage Method (string)

Creates NImage (see page 834) object from JPEG file.

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.

Returns

A NImage (see page 834) object.

6.2.5.1.2.2.2 SaveImage Method**6.2.5.1.2.2.2.1 Jpeg.SaveImage Method (NImage)**

Saves NImage (see page 834) to NBuffer (see page 886).

C#

```
public static NBuffer SaveImage(NImage image);
```

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.

Returns

NBuffer (see page 886) object.

6.2.5.1.2.2.2.2 Jpeg.SaveImage Method (NImage, Stream)

Saves NImage (see page 834) object to stream in JPEG format.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
Stream stream	The data stream used to save the image.

6.2.5.1.2.2.2.3 Jpeg.SaveImage Method (NImage, int)

Saves NImage (see page 834) object to byte array in JPEG format with specified quality.

C#

```
public static NBuffer SaveImage(NImage image, int quality);
```

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.

int quality	Specifies quality of JPEG image.

Returns

NBuffer ([see page 886](#)) object.

6.2.5.1.2.2.4 Jpeg.SaveImage Method (NImage, int, Stream)

Saves NImage ([see page 834](#)) object to stream in JPEG format with specified image quality.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
int quality	Specifies quality of JPEG image.
Stream stream	The data stream used to save the image.

6.2.5.1.2.2.5 Jpeg.SaveImage Method (NImage, int, string)

Saves NImage ([see page 834](#)) object to file in JPEG format with specified quality.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
int quality	Specifies quality of JPEG image.
string fileName	A string that contains the name of the file.

6.2.5.1.2.2.6 Jpeg.SaveImage Method (NImage, string)

Saves NImage ([see page 834](#)) object to file in JPEG format.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
string fileName	A string that contains the name of the file.

6.2.5.1.3 LosslessJpeg Class

Provides functionality for saving images in Lossless JPEG format.

Class Hierarchy**C#**

```
public static class LosslessJpeg;
```

LosslessJpeg Methods

	Name	Description
	SaveImage (see page 832)	Saves NImage (see page 834) to NBuffer (see page 886) in Lossless JPEG format.

6.2.5.1.3.1 LosslessJpeg Methods

6.2.5.1.3.1.1 SaveImage Method

6.2.5.1.3.1.1.1 LosslessJpeg.SaveImage Method (NImage)

Saves NImage (see page 834) to NBuffer (see page 886) in Lossless JPEG format.

C#

```
public static NBuffer SaveImage(NImage image);
```

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.

Returns

NBuffer (see page 886) object.

6.2.5.1.3.1.1.2 LosslessJpeg.SaveImage Method (NImage, Stream)

Saves NImage (see page 834) object to stream in Lossless JPEG format.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
Stream stream	The data stream used to save the image.

6.2.5.1.3.1.1.3 LosslessJpeg.SaveImage Method (NImage, string)

Saves NImage (see page 834) object to file in Lossless JPEG format.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
string fileName	A string that contains the name of the file.

6.2.5.1.4 NGrayscaleImage Class

Provides functionality for managing 8-bit grayscale images.

Class Hierarchy

C#

```
public sealed class NGrayscaleImage : NImage;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NImage Class

	Name	Description
	Clone (see page 835)	Creates NImage (see page 834) object from another NImage (see page 834) object.
	Copy (see page 836)	Copies specified rectangular area of the NImage (see page 834) to other NImage (see page 834) object.
	Create (see page 836)	Creates an image with specified pixel format, size, stride and resolution.
	Crop (see page 837)	Crops an image.
	FlipDiagonally (see page 838)	Flips image diagonally.
	FlipHorizontally (see page 838)	Flips image horizontally.
	FlipVertically (see page 838)	Flips image vertically.
	FromBitmap (see page 838)	Creates NImage (see page 834) from Bitmap.
	FromData (see page 838)	Creates NImage (see page 834) object from data with specified resolution.
	FromFile (see page 843)	Creates NImage (see page 834) object from file.
	FromHandle (see page 844)	Creates NImage (see page 834) object from specified handle.
	FromHBitmap (see page 844)	Creates NImage (see page 834) object from handle.
	FromImage (see page 844)	Creates NImage (see page 834) object from another NImage (see page 834) object.
	GetWrapper (see page 848)	Creates NImage (see page 834) object wrapper.
	RotateFlip (see page 854)	Rotates image using rotation type.
	Save (see page 854)	Saves NImage (see page 834) object to file.
	ToBitmap (see page 854)	Creates a Bitmap.
	ToHBitmap (see page 854)	Creates Windows HBITMAP.

NGrayscaleImage Class

	Name	Description
	this (see page 834)	Gets or sets the color of the specified pixel in NImage (see page 834) object.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NImage Class

	Name	Description
	Height (see page 855)	Gets height of image from NImage (see page 834) object.
	HorzResolution (see page 855)	Gets horizontal resolution in pixels per inch of image.

	LongSize (see page 855)	Gets size of NImage (see page 834) object.
	LongStride (see page 855)	Gets stride of image from NImage (see page 834) object.
	PixelFormat (see page 856)	Gets NPixelFormat (see page 870) of NImage (see page 834) object.
	Pixels (see page 856)	Gets pointer to array of pixels from NImage (see page 834) object.
	Size (see page 856)	Gets size of NImage (see page 834) object.
	Stride (see page 856)	Gets stride of image from NImage (see page 834) object.
	VertResolution (see page 857)	Gets vertical resolution in pixels per inch of image.
	Width (see page 857)	Gets width of image from NImage (see page 834) object.

6.2.5.1.4.1 NGrayscaleImage Methods

6.2.5.1.4.1.1 NGrayscaleImage.this Indexer

Gets or sets the color of the specified pixel in NImage ([see page 834](#)) object.

C#

```
public byte this[uint x, uint y];
```

Parameters

Parameters	Description
uint x	The x coordinate of the pixel.
uint y	The y coordinate of the pixel.

Returns

A color of specified pixel.

6.2.5.1.5 NImage Class

Provides functionality for managing images.

Class Hierarchy

C#

```
public class NImage : NObject, ICloneable;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NImage Class

	Name	Description
	Clone (see page 835)	Creates NImage object from another NImage object.
	Copy (see page 836)	Copies specified rectangular area of the NImage to other NImage object.
	Create (see page 836)	Creates an image with specified pixel format, size, stride and resolution.
	Crop (see page 837)	Crops an image.
	FlipDiagonally (see page 838)	Flips image diagonally.
	FlipHorizontally (see page 838)	Flips image horizontally.
	FlipVertically (see page 838)	Flips image vertically.
	FromBitmap (see page 838)	Creates NImage from Bitmap.
	FromData (see page 838)	Creates NImage object from data with specified resolution.
	FromFile (see page 843)	Creates NImage object from file.
	FromHandle (see page 844)	Creates NImage object from specified handle.
	FromHBitmap (see page 844)	Creates NImage object from handle.
	FromImage (see page 844)	Creates NImage object from another NImage object.
	GetWrapper (see page 848)	Creates NImage object wrapper.
	RotateFlip (see page 854)	Rotates image using rotation type.
	Save (see page 854)	Saves NImage object to file.
	ToBitmap (see page 854)	Creates a Bitmap.
	ToHBitmap (see page 854)	Creates Windows HBITMAP.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NImage Class

	Name	Description
	Height (see page 855)	Gets height of image from NImage object.
	HorzResolution (see page 855)	Gets horizontal resolution in pixels per inch of image.
	LongSize (see page 855)	Gets size of NImage object.
	LongStride (see page 855)	Gets stride of image from NImage object.
	PixelFormat (see page 856)	Gets NPixelFormat (see page 870) of NImage object.
	Pixels (see page 856)	Gets pointer to array of pixels from NImage object.
	Size (see page 856)	Gets size of NImage object.
	Stride (see page 856)	Gets stride of image from NImage object.
	VertResolution (see page 857)	Gets vertical resolution in pixels per inch of image.
	Width (see page 857)	Gets width of image from NImage object.

6.2.5.1.5.1 NImage Methods**6.2.5.1.5.1.1 NImage.Clone Method**

Creates NImage ([see page 834](#)) object from another NImage ([see page 834](#)) object.

C#

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

Returns

A NImage (see page 834) object.

6.2.5.1.5.1.2 NImage.Copy Method

Copies specified rectangular area of the NImage (see page 834) to other NImage (see page 834) object.

C#

```
public static void Copy(NImage srcImage, uint left, uint top, uint width, uint height,
NImage dstImage, uint dstLeft, uint dstTop);
```

Parameters

Parameters	Description
NImage srcImage	NImage (see page 834) object to copy.
uint left	The x coordinate of source rectangle.
uint top	The y coordinate of source rectangle.
uint width	The width of source rectangle.
uint height	The height of source rectangle.
NImage dstImage	A destination NImage (see page 834) object. Specified rectangular area will be copied to this object.
uint dstLeft	The x coordinate in the destination NImage (see page 834) object. Copied NImage (see page 834) will be placed starting by this coordinate.
uint dstTop	The y coordinate in the destination NImage (see page 834) object. Copied NImage (see page 834) will be placed starting by this coordinate.

Exceptions

Exceptions	Description
ArgumentNullException	srcImage or dstImage has null value.

6.2.5.1.5.1.3 Create Method

6.2.5.1.5.1.3.1 NImage.Create Method (NPixelFormat, uint, uint, uint, float, float)

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

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	Specifies pixel format of the image.
uint width	Specifies width of the image.
uint height	Specifies height of the image.
uint stride	Specifies stride of the image.
float horzResolution	Specifies horizontal resolution in pixels per inch of the image.
float vertResolution	Specifies vertical resolution in pixels per inch of the image.

Returns

If the function succeeds the return value is N_OK (see page 82).

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

- N_E_ARGUMENT (see page 78) - *pixelFormat* has invalid value.
- N_E_ARGUMENT (see page 78) - *stride* is not zero and is less than minimal value for specified pixel format and width.
- N_E_ARGUMENT_NULL (see page 78) - *pHImage* is NULL (see page 133).
- N_E_ARGUMENT_OUT_OF_RANGE (see page 78) - *width* or *height* is zero.
- N_E_ARGUMENT_OUT_OF_RANGE (see page 78) - *horzResolution* or *vertResolution* is less than zero.
- N_E_OUT_OF_MEMORY (see page 81) - there was not enough memory.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see *NImageGetStride* (see page 218) function. Created image must be deleted using *NImageFree* function. *horzResolution* and *vertResolution* can be zero if resolution is not applicable for the image.

6.2.5.1.5.1.3.2 NImage.Create Method (NPixelFormat, uint, uint, ulong, float, float)

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

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	Specifies pixel format of the image.
uint width	Specifies width of the image.
uint height	Specifies height of the image.
ulong stride	Specifies stride of the image.
float horzResolution	Specifies horizontal resolution in pixels per inch of the image.
float vertResolution	Specifies vertical resolution in pixels per inch of the image.

Returns

If the function succeeds the return value is N_OK (see page 82).

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

- N_E_ARGUMENT (see page 78) - *pixelFormat* has invalid value.
- N_E_ARGUMENT (see page 78) - *stride* is not zero and is less than minimal value for specified pixel format and width.
- N_E_ARGUMENT_NULL (see page 78) - *pHImage* is NULL (see page 133).
- N_E_ARGUMENT_OUT_OF_RANGE (see page 78) - *width* or *height* is zero.
- N_E_ARGUMENT_OUT_OF_RANGE (see page 78) - *horzResolution* or *vertResolution* is less than zero.
- N_E_OUT_OF_MEMORY (see page 81) - there was not enough memory.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see *NImageGetStride* (see page 218) function. Created image must be deleted using *NImageFree* function. *horzResolution* and *vertResolution* can be zero if resolution is not applicable for the image.

6.2.5.1.5.1.4 NImage.Crop Method

Crops an image.

C#

```
public NImage Crop(uint left, uint top, uint width, uint height);
```

Parameters

Parameters	Description
uint left	X-coordinate of the cropping rectangle.
uint top	Y-coordinate of the cropping rectangle.
uint width	Width (see page 857) of the cropping rectangle.
uint height	Height (see page 855) of the cropping rectangle.

Returns

Cropped NImage (see page 834) object.

6.2.5.1.5.1.5 NImage.FlipDiagonally Method

Flips image diagonally.

C#

```
public void FlipDiagonally();
```

6.2.5.1.5.1.6 NImage.FlipHorizontally Method

Flips image horizontally.

C#

```
public void FlipHorizontally();
```

6.2.5.1.5.1.7 NImage.FlipVertically Method

Flips image vertically.

C#

```
public void FlipVertically();
```

6.2.5.1.5.1.8 NImage.FromBitmap Method

Creates NImage (see page 834) from Bitmap.

C#

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

Parameters

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

Returns

A NImage (see page 834) object.

6.2.5.1.5.1.9 FromData Method**6.2.5.1.5.1.9.1 NImage.FromData Method (NPixelFormat, uint, uint, uint, float, float, int, byte[])**

Creates NImage (see page 834) object from data with specified resolution.

C#

```
public static NImage FromData(NPixelFormat pixelFormat, uint width, uint height, uint stride, float horzResolution, float vertResolution, int srcStride, byte[] srcPixels);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
int srcStride	A stride of source image.
byte[] srcPixels	A pointer to source pixel array.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

Format of memory block srcPixels points to must be the same as described in Pixels (see page 856) property, only stride is equal to srcStride.

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

6.2.5.1.5.1.9.2 NImage.FromData Method (NPixelFormat, uint, uint, uint, float, float, int, byte[], int, int)

Creates NImage (see page 834) object from data with specified resolution.

C#

```
public static NImage FromData(NPixelFormat pixelFormat, uint width, uint height, uint stride, float horzResolution, float vertResolution, int srcStride, byte[] srcPixels, int srcPixelsIndex, int srcPixelsCount);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
int srcStride	A stride of source image.
byte[] srcPixels	A pointer to source pixel array.
int srcPixelsIndex	Index in srcPixels byte array where image pixels data starts.
int srcPixelsCount	Number of pixels in byte array.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

Format of memory block srcPixels points to must be the same as described in Pixels (see page 856) property, only stride is equal to srcStride.

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

6.2.5.1.5.1.9.3 NImage.FromData Method (NPixelFormat, uint, uint, uint, float, float, int, byte[], int, int, uint, uint)

Creates NImage (see page 834) object from data with specified resolution.

C#

```
public static NImage FromData(NPixelFormat pixelFormat, uint width, uint height, uint stride, float horzResolution, float vertResolution, int srcStride, byte[] srcPixels, int srcPixelsIndex, int srcPixelsCount, uint left, uint top);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
int srcStride	A stride of source image.
byte[] srcPixels	A pointer to source pixel array.
int srcPixelsIndex	Index in srcPixels byte array where image pixels data starts.
int srcPixelsCount	Number of pixels in byte array.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

Format of memory block srcPixels points to must be the same as described in Pixels (see page 856) property, only stride is equal to srcStride.

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

6.2.5.1.5.1.9.4 NImage.FromData Method (NPixelFormat, uint, uint, uint, float, float, int, byte[], uint, uint)

Creates NImage (see page 834) object from data with specified resolution.

C#

```
public static NImage FromData(NPixelFormat pixelFormat, uint width, uint height, uint stride, float horzResolution, float vertResolution, int srcStride, byte[] srcPixels, uint left, uint top);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.

float vertResolution	A vertical resolution in pixels per inch of image.
int srcStride	A stride of source image.
byte[] srcPixels	A pointer to source pixel array.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

Format of memory block srcPixels points to must be the same as described in Pixels (see page 856) property, only stride is equal to srcStride.

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

6.2.5.1.5.1.9.5 NImage.FromData Method (NPixelFormat, uint, uint, uint, float, float, uint, IntPtr)

Creates NImage (see page 834) object from data with specified resolution.

C#

```
public static NImage FromData(NPixelFormat pixelFormat, uint width, uint height, uint stride, float horzResolution, float vertResolution, uint srcStride, IntPtr srcPixels);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
uint srcStride	A stride of source image.
IntPtr srcPixels	A pointer to source pixel array.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

Format of memory block srcPixels points to must be the same as described in Pixels (see page 856) property, only stride is equal to srcStride.

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

6.2.5.1.5.1.9.6 NImage.FromData Method (NPixelFormat, uint, uint, uint, float, float, uint, IntPtr, uint, uint)

Creates NImage (see page 834) object from data with specified resolution.

C#

```
public static NImage FromData(NPixelFormat pixelFormat, uint width, uint height, uint stride, float horzResolution, float vertResolution, uint srcStride, IntPtr srcPixels, uint left, uint top);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
uint srcStride	A stride of source image.
IntPtr srcPixels	A pointer to source pixel array.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

Format of memory block srcPixels points to must be the same as described in Pixels (see page 856) property, only stride is equal to srcStride.

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

6.2.5.1.5.1.9.7 NImage.FromData Method (NPixelFormat, uint, uint, ulong, float, float, ulong, IntPtr)

Creates NImage (see page 834) object from data with specified resolution.

C#

```
public static NImage FromData(NPixelFormat pixelFormat, uint width, uint height, ulong stride, float horzResolution, float vertResolution, ulong srcStride, IntPtr srcPixels);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
ulong stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
ulong srcStride	A stride of source image.
IntPtr srcPixels	A pointer to source pixel array.

Returns

A NImage (see page 834) object.

6.2.5.1.5.1.9.8 NImage.FromData Method (NPixelFormat, uint, uint, ulong, float, float, ulong, IntPtr, uint, uint)

Creates NImage (see page 834) object from data with specified resolution.

C#

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

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
ulong stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
ulong srcStride	A stride of source image.
IntPtr srcPixels	A pointer to source pixel array.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

Format of memory block srcPixels points to must be the same as described in Pixels (see page 856) property, only stride is equal to srcStride.

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

6.2.5.1.5.1.10 FromFile Method

6.2.5.1.5.1.10.1 NImage.FromFile Method (string)

Creates NImage (see page 834) object from file.

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.

Returns

A NImage (see page 834) object.

6.2.5.1.5.1.10.2 NImage.FromFile Method (string, NImageFormat)

Creates NImage (see page 834) object from file with specified NImageFormat (see page 859).

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.
NImageFormat imageFormat	An image NImageFormat (see page 859) object.

Returns

A NImage (see page 834) object.

6.2.5.1.5.1.11 FromHandle Method**6.2.5.1.5.1.11.1 NImage.FromHandle Method (IntPtr)**

Creates NImage (see page 834) object from specified handle.

C#

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

Parameters

Parameters	Description
IntPtr handle	Pointer to NImage (see page 834) handle.

Returns

Created NImage (see page 834) object.

6.2.5.1.5.1.11.2 NImage.FromHandle Method (IntPtr, bool)

Creates NImage (see page 834) object from specified handle.

C#

```
public static NImage FromHandle(IntPtr handle, bool ownsHandle);
```

Parameters

Parameters	Description
IntPtr handle	Pointer to NImage (see page 834) handle.
bool ownsHandle	If set to true, on destruction of the NImage (see page 834) the NImage (see page 834) pointed to by the handle will also be destroyed.

Returns

Created NImage (see page 834) object.

6.2.5.1.5.1.12 NImage.FromHBitmap Method

Creates NImage (see page 834) object from handle.

C#

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

Parameters

Parameters	Description
IntPtr hBitmap	A pointer to handle.

Returns

A NImage (see page 834) object.

6.2.5.1.5.1.13 FromImage Method**6.2.5.1.5.1.13.1 NImage.FromImage Method (NPixelFormat, uint, NImage)**

Creates NImage (see page 834) object from another NImage (see page 834) object.

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint stride	A stride of image.
NImage srcImage	A NImage (see page 834) source object.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

6.2.5.1.5.1.13.2 NImage.FromImage Method (NPixelFormat, uint, NImage, uint, uint, uint, uint)

Creates NImage (see page 834) object from another NImage (see page 834) object.

C#

```
public static NImage FromImage(NPixelFormat pixelFormat, uint stride, NImage srcImage, uint left, uint top, uint width, uint height);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint stride	A stride of image.
NImage srcImage	A NImage (see page 834) source object.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.
uint width	A width of image.
uint height	A height of image.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

6.2.5.1.5.1.13.3 NImage.FromImage Method (NPixelFormat, uint, float, float, NImage)

Creates NImage (see page 834) object from another NImage (see page 834) object.

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.

float vertResolution	A vertical resolution in pixels per inch of image.
NImage srcImage	A NImage (see page 834) source object.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

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

6.2.5.1.5.1.13.4 NImage.FromImage Method (NPixelFormat, uint, float, float, NImage, uint, uint, uint, uint)

Creates NImage (see page 834) object from another NImage (see page 834) object.

C#

```
public static NImage FromImage(NPixelFormat pixelFormat, uint stride, float horzResolution, float vertResolution, NImage srcImage, uint left, uint top, uint width, uint height);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
NImage srcImage	A NImage (see page 834) source object.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.
uint width	A width of image.
uint height	A height of image.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

6.2.5.1.5.1.13.5 NImage.FromImage Method (NPixelFormat, ulong, NImage)

Creates NImage (see page 834) object from another NImage (see page 834) object.

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
ulong stride	A stride of image.
NImage srcImage	A NImage (see page 834) source object.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

6.2.5.1.5.1.13.6 NImage.FromImage Method (NPixelFormat, ulong, NImage, uint, uint, uint, uint)

Creates NImage (see page 834) object from another NImage (see page 834) object.

C#

```
public static NImage FromImage(NPixelFormat pixelFormat, ulong stride, NImage srcImage,
    uint left, uint top, uint width, uint height);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
ulong stride	A stride of image.
NImage srcImage	A NImage (see page 834) source object.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.
uint width	A width of image.
uint height	A height of image.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

6.2.5.1.5.1.13.7 NImage.FromImage Method (NPixelFormat, ulong, float, float, NImage)

Creates NImage (see page 834) object from another NImage (see page 834) object.

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
ulong stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
NImage srcImage	A NImage (see page 834) source object.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

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

6.2.5.1.5.1.13.8 NImage.FromImage Method (NPixelFormat, ulong, float, float, NImage, uint, uint, uint, uint)

Creates NImage (see page 834) object from another NImage (see page 834) object.

C#

```
public static NImage FromImage(NPixelFormat pixelFormat, ulong stride, float horzResolution, float vertResolution, NImage srcImage, uint left, uint top, uint width, uint height);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
ulong stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
NImage srcImage	A NImage (see page 834) source object.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.
uint width	A width of image.
uint height	A height of image.

Returns

A NImage (see page 834) object.

Remarks

If stride is zero then image stride is automatically calculated. For more information on image stride see Stride (see page 856) property.

6.2.5.1.5.1.14 GetWrapper Method

6.2.5.1.5.1.14.1 NImage.GetWrapper Method (NImage, uint, uint, uint, uint)

Creates NImage (see page 834) object wrapper.

C#

```
public static NImage GetWrapper(NImage srcImage, uint left, uint top, uint width, uint height);
```

Parameters

Parameters	Description
NImage srcImage	A NImage (see page 834) source object.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.
uint width	A width of image.
uint height	A height of image.

Returns

A NImage (see page 834) object.

Remarks

For more information on image stride see Stride (see page 856) property.

Format of memory block pixels points to must be the same as described in Pixels (see page 856) property.

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

Exceptions

Exceptions	Description
ArgumentNullException	srcImage is null.

6.2.5.1.5.1.14.2 NImage.GetWrapper Method (NImage, uint, uint, uint, uint, float, float)

Creates NImage (see page 834) object wrapper.

C#

```
public static NImage GetWrapper(NImage srcImage, uint left, uint top, uint width, uint height, float horzResolution, float vertResolution);
```

Parameters

Parameters	Description
NImage srcImage	A NImage (see page 834) source object.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.
uint width	A width of image.
uint height	A height of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.

Returns

A NImage (see page 834) object.

Remarks

For more information on image stride see Stride (see page 856) property.

Format of memory block pixels points to must be the same as described in Pixels (see page 856) property.

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

6.2.5.1.5.1.14.3 NImage.GetWrapper Method (NPixelFormat, uint, uint, int, float, float, byte[])

Creates NImage (see page 834) object wrapper.

C#

```
public static NImage GetWrapper(NPixelFormat pixelFormat, uint width, uint height, int stride, float horzResolution, float vertResolution, byte[] srcPixels);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
int stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
byte[] srcPixels	A source pixel array.

Returns

A NImage (see page 834) object.

Remarks

For more information on image stride see Stride (see page 856) property.

Format of memory block pixels points to must be the same as described in Pixels (see page 856) property.

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

6.2.5.1.5.1.14.4 NImage.GetWrapper Method (NPixelFormat, uint, uint, int, float, float, byte[], int, int)

Creates NImage (see page 834) object wrapper.

C#

```
public static NImage GetWrapper(NPixelFormat pixelFormat, uint width, uint height, int stride, float horzResolution, float vertResolution, byte[] pixels, int pixelsIndex, int pixelsCount);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
int stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
byte[] pixels	A source pixel array.
int pixelsIndex	Index in srcPixels byte array where image pixels data starts.
int pixelsCount	Number of pixels in byte array.

Returns

A NImage (see page 834) object.

Remarks

For more information on image stride see Stride (see page 856) property.

Format of memory block pixels points to must be the same as described in Pixels (see page 856) property.

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

6.2.5.1.5.1.14.5 NImage.GetWrapper Method (NPixelFormat, uint, uint, int, float, float, byte[], int, int, uint, uint)

Creates NImage (see page 834) object wrapper.

C#

```
public static NImage GetWrapper(NPixelFormat pixelFormat, uint width, uint height, int stride, float horzResolution, float vertResolution, byte[] pixels, int pixelsIndex, int pixelsCount, uint left, uint top);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
int stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
byte[] pixels	A source pixel array.
int pixelsIndex	Index in srcPixels byte array where image pixels data starts.
int pixelsCount	Number of pixels in byte array.

uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.

Returns

A *NImage* (see page 834) object.

Remarks

For more information on image stride see *Stride* (see page 856) property.

Format of memory block pixels points to must be the same as described in *Pixels* (see page 856) property.

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

6.2.5.1.5.1.14.6 NImage.GetWrapper Method (NPixelFormat, uint, uint, int, float, float, byte[], uint, uint)

Creates *NImage* (see page 834) object wrapper.

C#

```
public static NImage GetWrapper(NPixelFormat pixelFormat, uint width, uint height, int stride, float horzResolution, float vertResolution, byte[] srcPixels, uint left, uint top);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	A <i>NPixelFormat</i> (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
int stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
byte[] srcPixels	A source pixel array.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.

Returns

A *NImage* (see page 834) object.

Remarks

For more information on image stride see *Stride* (see page 856) property.

Format of memory block pixels points to must be the same as described in *Pixels* (see page 856) property.

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

6.2.5.1.5.1.14.7 NImage.GetWrapper Method (NPixelFormat, uint, uint, uint, float, float, IntPtr, bool)

Creates *NImage* (see page 834) object wrapper.

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A <i>NPixelFormat</i> (see page 870) of image.
uint width	A width of image.
uint height	A height of image.

uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
IntPtr pixels	Pointer to memory block containing pixels for the image.
bool ownsPixels	Specifies whether pixels will be automatically deleted with the image (if set to true).

Returns

A *NImage* (see page 834) object.

Remarks

For more information on image stride see *Stride* (see page 856) property.

Format of memory block pixels points to must be the same as described in *Pixels* (see page 856) property.

pixels must not be deleted during lifetime of the image. If *ownsPixels* is true then pixels will be automatically deleted with the image.

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

6.2.5.1.5.1.14.8 NImage.GetWrapper Method (NPixelFormat, uint, uint, uint, float, float, IntPtr, uint, uint)

Creates *NImage* (see page 834) object wrapper.

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A <i>NPixelFormat</i> (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
uint stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
IntPtr pixels	Pointer to memory block which contains image data.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.

Returns

A *NImage* (see page 834) object.

Remarks

For more information on image stride see *Stride* (see page 856) property.

Format of memory block pixels points to must be the same as described in *Pixels* (see page 856) property.

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

6.2.5.1.5.1.14.9 NImage.GetWrapper Method (NPixelFormat, uint, uint, ulong, float, float, IntPtr, bool)

Creates *NImage* (see page 834) object wrapper.

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
ulong stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
IntPtr pixels	Pointer to memory block containing pixels for the image.
bool ownsPixels	Specifies whether pixels will be automatically deleted with the image (if set to true).

Returns

A NImage (see page 834) object.

Remarks

For more information on image stride see Stride (see page 856) property.

Format of memory block pixels points to must be the same as described in Pixels (see page 856) property.

pixels must not be deleted during lifetime of the image. If *ownsPixels* is true then pixels will be automatically deleted with the image.

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

6.2.5.1.5.1.14.10 NImage.GetWrapper Method (NPixelFormat, uint, uint, ulong, float, float, IntPtr, uint, uint)

Creates NImage (see page 834) object wrapper.

C#

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

Parameters

Parameters	Description
NPixelFormat pixelFormat	A NPixelFormat (see page 870) of image.
uint width	A width of image.
uint height	A height of image.
ulong stride	A stride of image.
float horzResolution	A horizontal resolution in pixels per inch of image.
float vertResolution	A vertical resolution in pixels per inch of image.
IntPtr pixels	Pointer to memory block containing pixels for the image.
uint left	X-coordinate of the image rectangle.
uint top	Y-coordinate of the image rectangle.

Returns

A NImage (see page 834) object.

Remarks

For more information on image stride see Stride (see page 856) property.

Format of memory block pixels points to must be the same as described in Pixels (see page 856) property.

pixels must not be deleted during lifetime of the image. If *ownsPixels* is true then pixels will be automatically deleted with the image.

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

6.2.5.1.5.1.15 NImage.RotateFlip Method

Rotates image using rotation type.

C#

```
public NImage RotateFlip(NImageRotateFlipType rotateFlipType);
```

Parameters

Parameters	Description
NImageRotateFlipType rotateFlipType	One of the rotation type value.

Returns

Rotated NImage (see page 834) object.

6.2.5.1.5.1.16 Save Method

6.2.5.1.5.1.16.1 NImage.Save Method (string)

Saves NImage (see page 834) object to file.

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.

6.2.5.1.5.1.16.2 NImage.Save Method (string, NImageFormat)

Saves NImage (see page 834) object to file with specified NImageFormat (see page 859).

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.
NImageFormat imageFormat	An image NImageFormat (see page 859) object.

6.2.5.1.5.1.17 NImage.ToBitmap Method

Creates a Bitmap.

C#

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

Returns

A Bitmap object.

6.2.5.1.5.1.18 NImage.ToHBitmap Method

Creates Windows HBITMAP.

C#

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

Returns

A Windows HBITMAP.

6.2.5.1.5.2 NImage Properties

6.2.5.1.5.2.1 NImage.Height Property

Gets height of image from NImage (see page 834) object.

C#

```
public uint Height;
```

Property value

A height of image.

6.2.5.1.5.2.2 NImage.HorzResolution Property

Gets horizontal resolution in pixels per inch of image.

C#

```
public float HorzResolution;
```

Remarks

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

Property value

A horizontal resolution in pixels per inch of image.

6.2.5.1.5.2.3 NImage.LongSize Property

Gets size of NImage (see page 834) object.

C#

```
public ulong LongSize;
```

Remarks

Size (see page 856) of memory block containing image pixels is equal to image height multiplied by image stride. For more information see Height (see page 855) and Stride (see page 856) properties.

Property value

A size of NImage (see page 834) object.

6.2.5.1.5.2.4 NImage.LongStride Property

Gets stride of image from NImage (see page 834) object.

C#

```
public ulong LongStride;
```

Remarks

Stride (see page 856) (size of one row) of the image depends on image pixel format and width. It cannot be less than value obtained with GetRowLongSize or GetRowSize methods with arguments obtained with PixelFormat (see page 856) and Width (see page 857) properties.

Property value

A stride of image.

6.2.5.1.5.2.5 NImage.PixelFormat Property

Gets NPixelFormat (see page 870) of NImage (see page 834) object.

C#

```
public NPixelFormat PixelFormat;
```

Property value

A NPixelFormat (see page 870) structure.

6.2.5.1.5.2.6 NImage.Pixels Property

Gets pointer to array of pixels from NImage (see page 834) object.

C#

```
public IntPtr Pixels;
```

Remarks

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

For more information see PixelFormat (see page 856), Width (see page 857), Height (see page 855), Stride (see page 856), and Size (see page 856) properties.

Property value

A pointer to pixel array.

6.2.5.1.5.2.7 NImage.Size Property

Gets size of NImage (see page 834) object.

C#

```
public uint Size;
```

Remarks

Size of memory block containing image pixels is equal to image height multiplied by image stride. For more information see Height (see page 855) and Stride (see page 856) properties.

Property value

A size of NImage (see page 834) object.

6.2.5.1.5.2.8 NImage.Stride Property

Gets stride of image from NImage (see page 834) object.

C#

```
public uint Stride;
```

Remarks

Stride (size of one row) of the image depends on image pixel format and width. It cannot be less than value obtained with GetRowLongSize or GetRowSize methods with arguments obtained with PixelFormat (see page 856) and Width (see page 857) properties.

Property value

A stride of image.

6.2.5.1.5.2.9 NImage.VertResolution Property

Gets vertical resolution in pixels per inch of image.

C#

```
public float VertResolution;
```

Remarks

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

Property value

A vertical resolution in pixels per inch of image.

6.2.5.1.5.2.10 NImage.Width Property

Gets width of image from NImage (see page 834) object.

C#

```
public uint Width;
```

Property value

A width of image.

6.2.5.1.6 NImageFile Class

Provides functionality for working with image files.

Class Hierarchy

C#

```
public class NImageFile : NObject;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NImageFile Class

	Name	Description
	Close (see page 858)	Closes opened image file.
	FromFile (see page 858)	Creates (loads) an image from specified file.
	ReadImage (see page 858)	Reads image file.

NObject Properties

NObject Class

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NImageFile Class

	Name	Description
	Format (see page 859)	Gets format of image within NImage (see page 834) object.

6.2.5.1.6.1 NImageFile Methods

6.2.5.1.6.1.1 NImageFile.Close Method

Closes opened image file.

C#

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

6.2.5.1.6.1.2 FromFile Method

6.2.5.1.6.1.2.1 NImageFile.FromFile Method (string)

Creates (loads) an image from specified file.

C#

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

Parameters

Parameters	Description
string fileName	String which specified file name.

Returns

NImageFile ([see page 857](#)) object which contains loaded image.

6.2.5.1.6.1.2.2 NImageFile.FromFile Method (string, NImageFormat)

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

C#

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

Parameters

Parameters	Description
string fileName	String which specified file name.
NImageFormat imageFormat	File image format.

Returns

NImageFile ([see page 857](#)) object which contains loaded image.

6.2.5.1.6.1.3 NImageFile.ReadImage Method

Reads image file.

C#

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

Returns

A NImage (see page 834) object.

6.2.5.1.6.2 NImageFile Properties**6.2.5.1.6.2.1 NImageFile.Format Property**

Gets format of image within NImage (see page 834) object.

C#

```
public virtual NImageFormat Format;
```

Property value

Format of an image within NImage (see page 834).

6.2.5.1.7 NImageFormat Class

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

Class Hierarchy

C#

```
public class NImageFormat : NObject;
```

NImageFormat Classes**NImageFormat Class**

	Name	Description
	ImageFormatCollection (see page 860)	Represents the collection of formats in a NImageFormat.

NImageFormat Fields**NImageFormat Class**

	Name	Description
	Bmp (see page 861)	Specifies the BMP image format.
	Formats (see page 861)	Specifies collection of supported image formats.
	Gif (see page 861)	Specifies the GIF image format.
	IHead (see page 861)	Specifies the NIST IHead image format.
	Jpeg (see page 861)	Specifies the JPEG image format.
	Jpeg2K (see page 861)	Specifies the JPEG2000 image format.
	Png (see page 861)	Specifies the PNG image format.
	Tiff (see page 861)	Specifies the TIFF image format.
	Wsq (see page 861)	Specifies the WSQ image format.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NImageFormat Class

	Name	Description
	LoadImage (see page 862)	Loads NImage (see page 834) from memory buffer.
	OpenFile (see page 863)	Opens image file from memory buffer.
	SaveImage (see page 864)	Saves NImage (see page 834) to byte array.
	SaveImages (see page 865)	Saves specified number of images to the byte array.
	Select (see page 865)	Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NImageFormat Class

	Name	Description
	CanRead (see page 866)	Gets a value indicating whether the current image format supports reading.
	CanWrite (see page 866)	Gets a value indicating whether the current image format supports writing.
	CanWriteMultiple (see page 866)	Retrieves a value indicating whether the image format supports writing of multiple images.
	DefaultFileExtension (see page 866)	Gets default file extension of the current image format.
	FileFilter (see page 866)	Gets file filter of the current image format.
	Name (see page 866)	Gets name of the current image format.

6.2.5.1.7.1 NImageFormat Classes**6.2.5.1.7.1.1 NImageFormat.ImageFormatCollection Class**

Represents the collection of formats in a NImageFormat ([see page 859](#)).

Class Hierarchy**C#**

```
public sealed class ImageFormatCollection : NObjectStaticReadOnlyCollection<NImageFormat>;
```

6.2.5.1.7.2 NImageFormat Fields

6.2.5.1.7.2.1 NImageFormat.Bmp Field

Specifies the BMP image format.

C#

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

6.2.5.1.7.2.2 NImageFormat.Formats Field

Specifies collection of supported image formats.

C#

```
public static readonly ImageFormatCollection Formats;
```

6.2.5.1.7.2.3 NImageFormat.Gif Field

Specifies the GIF image format.

C#

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

6.2.5.1.7.2.4 NImageFormat.IHead Field

Specifies the NIST IHead image format.

C#

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

6.2.5.1.7.2.5 NImageFormat.Jpeg Field

Specifies the JPEG image format.

C#

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

6.2.5.1.7.2.6 NImageFormat.Jpeg2K Field

Specifies the JPEG2000 image format.

C#

```
public static readonly NImageFormat Jpeg2K;
```

6.2.5.1.7.2.7 NImageFormat.Png Field

Specifies the PNG image format.

C#

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

6.2.5.1.7.2.8 NImageFormat.Tiff Field

Specifies the TIFF image format.

C#

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

6.2.5.1.7.2.9 NImageFormat.Wsq Field

Specifies the WSQ image format.

C#

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

6.2.5.1.7.3 NImageFormat Methods

6.2.5.1.7.3.1 LoadImage Method

6.2.5.1.7.3.1.1 NImageFormat.LoadImage Method (IntPtr, int)

Loads NImage (see page 834) from memory buffer.

C#

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

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer.
int bufferLength	Size of memory buffer.

Returns

A NImage (see page 834) object.

6.2.5.1.7.3.1.2 NImageFormat.LoadImage Method (Stream)

Loads NImage (see page 834) from stream.

C#

```
public virtual NImage LoadImage(Stream stream);
```

Parameters

Parameters	Description
Stream stream	A sequence of bytes (stream)

Returns

A NImage (see page 834) object.

6.2.5.1.7.3.1.3 NImageFormat.LoadImage Method (byte[])

Loads NImage (see page 834) from byte array.

C#

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

Parameters

Parameters	Description
byte[] buffer	Pointer to memory buffer.

Returns

A NImage (see page 834) object.

6.2.5.1.7.3.1.4 NImageFormat.LoadImage Method (string)

Loads NImage (see page 834) from file.

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.

Returns

A NImage (see page 834) object.

6.2.5.1.7.3.2 OpenFile Method**6.2.5.1.7.3.2.1 NImageFormat.OpenFile Method (IntPtr, int)**

Opens image file from memory buffer.

C#

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

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer.
int bufferLength	Size of memory buffer.

Returns

NImage (see page 834) object.

6.2.5.1.7.3.2.2 NImageFormat.OpenFile Method (Stream, bool)

Opens image file from memory stream.

C#

```
public virtual NImageFile OpenFile(Stream stream, bool ownsStream);
```

Parameters

Parameters	Description
Stream stream	Memory stream.
bool ownsStream	If true, the stream is closed by the writer when done; otherwise false.

Returns

NImage (see page 834) object.

6.2.5.1.7.3.2.3 NImageFormat.OpenFile Method (byte[])

Opens image file from memory buffer.

C#

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

Parameters

Parameters	Description
byte[] buffer	Pointer to memory buffer.

Returns

NImage (see page 834) object.

6.2.5.1.7.3.2.4 NImageFormat.OpenFile Method (string)

Opens image file from a file.

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.

Returns

NImage ([see page 834](#)) object.

6.2.5.1.7.3.3 SaveImage Method

6.2.5.1.7.3.3.1 NImageFormat.SaveImage Method (NImage)

Saves NImage ([see page 834](#)) to byte array.

C#

```
public virtual NBuffer SaveImage(NImage image);
```

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.

Returns

A NBuffer ([see page 886](#)) object.

6.2.5.1.7.3.3.2 NImageFormat.SaveImage Method (NImage, Stream)

Saves NImage ([see page 834](#)) to stream.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
Stream stream	The data stream used to save the image.

6.2.5.1.7.3.3.3 NImageFormat.SaveImage Method (NImage, string)

Saves NImage ([see page 834](#)) to file.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
string fileName	A string that contains the name of the file.

6.2.5.1.7.3.4 SaveImages Method

6.2.5.1.7.3.4.1 NImageFormat.SaveImages Method (NImage[])

Saves specified number of images to the byte array.

C#

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

Parameters

Parameters	Description
NImage[] images	Array of NImage (see page 834) objects.

Returns

A NBuffer (see page 886) object.

6.2.5.1.7.3.4.2 NImageFormat.SaveImages Method (NImage[], Stream)

Saves specified number of images to the stream.

C#

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

Parameters

Parameters	Description
NImage[] images	Array of NImage (see page 834) objects.
Stream stream	The data stream used to save the images.

6.2.5.1.7.3.4.3 NImageFormat.SaveImages Method (NImage[], string)

Saves specified number of images to the files.

C#

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

Parameters

Parameters	Description
NImage[] images	Array of NImage (see page 834) objects.
string fileName	A string that contains the names of the files.

6.2.5.1.7.3.5 NImageFormat.Select Method

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

C#

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

Parameters

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

Returns

A NImageFormat (see page 859) object.

6.2.5.1.7.4 NImageFormat Properties

6.2.5.1.7.4.1 NImageFormat.CanRead Property

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

C#

```
public virtual bool CanRead;
```

Property value

true if image format can read, *false* if image format cannot read.

6.2.5.1.7.4.2 NImageFormat.CanWrite Property

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

C#

```
public virtual bool CanWrite;
```

Property value

true if image format can write, *false* if image format cannot write.

6.2.5.1.7.4.3 NImageFormat.CanWriteMultiple Property

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

C#

```
public virtual bool CanWriteMultiple;
```

Property value

true if image format can write, *false* if image format cannot write.

6.2.5.1.7.4.4 NImageFormat.DefaultFileExtension Property

Gets default file extension of the current image format.

C#

```
public virtual string DefaultFileExtension;
```

Property value

Default file extension.

6.2.5.1.7.4.5 NImageFormat.FileFilter Property

Gets file filter of the current image format.

C#

```
public virtual string FileFilter;
```

Property value

An image format file filter.

6.2.5.1.7.4.6 NImageFormat.Name Property

Gets name of the current image format.

C#

```
public virtual string Name;
```

Property value

An image format name.

6.2.5.1.8 NImages Class

Provides library registration and other additional functionality.

Class Hierarchy

C#

```
public static class NImages;
```

NImages Fields

	Name	Description
	DllName (see page 867)	Name of DLL containing unmanaged part of this class.

NImages Methods

	Name	Description
	GetGrayscaleColorWrapper (see page 867)	Creates NImage (see page 834) object wrapper.
	GetInfo (see page 868)	Retrieves information about NImages (see page 188) library.
	GetOpenFileFilter (see page 868)	Gets a list with names of all NImageFormats, which support reading. Names in the list are separated by semicolon.
	GetOpenFileFilterString (see page 868)	Gets a list with names of all NImageFormats, which support reading. Names in the list are separated by semicolon.
	GetSaveFileFilter (see page 868)	Gets a list with names of all NImageFormats, which support writing. Names in the list are separated by semicolon.
	GetSaveFileFilterString (see page 868)	Gets a list with names of all NImageFormats, which support writing. Names in the list are separated by semicolon.

6.2.5.1.8.1 NImages Fields

6.2.5.1.8.1.1 NImages.DllName Field

Name of DLL containing unmanaged part of this class.

C#

```
public const string DllName = "NImages";
```

6.2.5.1.8.2 NImages Methods

6.2.5.1.8.2.1 NImages.GetGrayscaleColorWrapper Method

Creates NImage ([see page 834](#)) object wrapper.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
NRgb minColor	Specifies color to be used for black color.
NRgb maxColor	Specifies color to be used for white color.

Returns

An NImage ([see page 834](#)) object.

Remarks

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

6.2.5.1.8.2.2 NImages.GetInfo Method

Retrieves information about NImages (see page 188) library.

C#

```
public static NLibraryInfo GetInfo();
```

Returns

NLibraryInfo (see page 619) object which contains information about library such as product name, serial number, version.

6.2.5.1.8.2.3 NImages.GetOpenFileFilter Method

Gets a list with names of all NImageFormats, which support reading. Names in the list are separated by semicolon.

C#

```
public static string GetOpenFileFilter();
```

Returns

String with NImageFormat (see page 859) names separated by semicolon.

6.2.5.1.8.2.4 NImages.GetOpenFileFilterString Method

Gets a list with names of all NImageFormats, which support reading. Names in the list are separated by semicolon.

C#

```
public static string GetOpenFileFilterString(bool addAllImage, bool addAllFiles);
```

Parameters

Parameters	Description
bool addAllImage	Boolean indicating whether to add "All image files" to the list.
bool addAllFiles	Boolean indicating whether to add "All files" to the list.

Returns

String with NImageFormat (see page 859) names separated by semicolon.

6.2.5.1.8.2.5 NImages.GetSaveFileFilter Method

Gets a list with names of all NImageFormats, which support writing. Names in the list are separated by semicolon.

C#

```
public static string GetSaveFileFilter();
```

Returns

String with NImageFormat (see page 859) names separated by semicolon.

6.2.5.1.8.2.6 NImages.GetSaveFileFilterString Method

Gets a list with names of all NImageFormats, which support writing. Names in the list are separated by semicolon.

C#

```
public static string GetSaveFileFilterString();
```

Returns

String with NImageFormat (see page 859) names separated by semicolon.

6.2.5.1.9 NMonochromeImage Class

Provides functionality for managing 1-bit monochrome images.

Class Hierarchy

C#

```
public sealed class NMonochromeImage : NImage;
```

Remarks

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

NDisposable Methods

	Name	Description
✚	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
✚	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
✚	Free (see page 621)	Releases memory resources used by specified object.
✚	GetNativeType (see page 622)	Retrieves native type of object.
✚	GetParameter (see page 622)	Gets value of parameter by parameter id.
✚	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
✚	SetParameter (see page 623)	Sets value of parameter by parameter id.

NImage Class

	Name	Description
✚	Clone (see page 835)	Creates NImage (see page 834) object from another NImage (see page 834) object.
✚	Copy (see page 836)	Copies specified rectangular area of the NImage (see page 834) to other NImage (see page 834) object.
✚	Create (see page 836)	Creates an image with specified pixel format, size, stride and resolution.
✚	Crop (see page 837)	Crops an image.
✚	FlipDiagonally (see page 838)	Flips image diagonally.
✚	FlipHorizontally (see page 838)	Flips image horizontally.
✚	FlipVertically (see page 838)	Flips image vertically.
✚	FromBitmap (see page 838)	Creates NImage (see page 834) from Bitmap.
✚	FromData (see page 838)	Creates NImage (see page 834) object from data with specified resolution.
✚	FromFile (see page 843)	Creates NImage (see page 834) object from file.
✚	FromHandle (see page 844)	Creates NImage (see page 834) object from specified handle.
✚	FromHBitmap (see page 844)	Creates NImage (see page 834) object from handle.
✚	FromImage (see page 844)	Creates NImage (see page 834) object from another NImage (see page 834) object.
✚	GetWrapper (see page 848)	Creates NImage (see page 834) object wrapper.
✚	RotateFlip (see page 854)	Rotates image using rotation type.
✚	Save (see page 854)	Saves NImage (see page 834) object to file.
✚	ToBitmap (see page 854)	Creates a Bitmap.
✚	ToHBitmap (see page 854)	Creates Windows HBITMAP.

NMonochromeImage Class

	Name	Description
	this (see page 870)	Gets or sets value of NMonochromeImage image pixel with specified coordinates.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NImage Class

	Name	Description
	Height (see page 855)	Gets height of image from NImage (see page 834) object.
	HorzResolution (see page 855)	Gets horizontal resolution in pixels per inch of image.
	LongSize (see page 855)	Gets size of NImage (see page 834) object.
	LongStride (see page 855)	Gets stride of image from NImage (see page 834) object.
	PixelFormat (see page 856)	Gets NPixelFormat (see page 870) of NImage (see page 834) object.
	Pixels (see page 856)	Gets pointer to array of pixels from NImage (see page 834) object.
	Size (see page 856)	Gets size of NImage (see page 834) object.
	Stride (see page 856)	Gets stride of image from NImage (see page 834) object.
	VertResolution (see page 857)	Gets vertical resolution in pixels per inch of image.
	Width (see page 857)	Gets width of image from NImage (see page 834) object.

6.2.5.1.9.1 NMonochromeImage Methods**6.2.5.1.9.1.1 NMonochromeImage.this Indexer**

Gets or sets value of NMonochromeImage (see page 869) image pixel with specified coordinates.

C#

```
public bool this[uint x, uint y];
```

Parameters

Parameters	Description
uint x	X coordinate of the pixel.
uint y	Y coordinate of the pixel.

Returns

Value of pixel, either 0 or 1.

6.2.5.1.10 NPixelFormat Structure

Provides functionality for working with pixel format.

C#

```
[StructLayout(LayoutKind.Sequential)]
public struct NPixelFormat {
    public static readonly NPixelFormat Monochrome = new NPixelFormat(0x00001001);
    public static readonly NPixelFormat Grayscale = new NPixelFormat(0x00301001);
    public static readonly NPixelFormat Rgb = new NPixelFormat(0x00303003);
}
```

Remarks

Image pixel format is not limited to these fields. However only these fields are provided for usage with this SDK.

NPixelFormat Fields

	Name	Description
	Grayscale (see page 871)	Each pixel value is stored in 8 bits representing 256 shades of gray.
	Monochrome (see page 871)	Each pixel value is stored in 1 bit representing either black (value 0) or white (value 1).
	Rgb (see page 872)	Each pixel value is stored in 24 bits consisting of three 8-bit values representing red, green and blue color components.

NPixelFormat Methods

	Name	Description
	CalcRowLongSize (see page 872)	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
	CalcRowSize (see page 872)	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
	Equals (see page 873)	Determines whether the specified Object is equal to the current Object.
	GetHashCode (see page 873)	Gets the hash code. Is intended for a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.
	GetRowLongSize (see page 873)	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.
	GetRowSize (see page 874)	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.
	IsValid (see page 875)	Checks whether current NPixelFormat value is valid.
	ToString (see page 875)	Retrieves string representation of NPixelFormat object.

NPixelFormat Operators

	Name	Description
	!= (see page 875)	Checks if two pixelFormats are not equal.
	== (see page 875)	Checks if two pixelFormats are equal.
	NPixelFormat (see page 876)	This conversion operator converts from a uint (see page 876) type to a NPixelFormat type
	uint (see page 876)	Gets value field of NPixelFormat object.

NPixelFormat Properties

	Name	Description
	BitsPerPixel (see page 876)	Each pixel value is stored in 24 bits consisting of three 8-bit values representing red, green and blue color components.

6.2.5.1.10.1 NPixelFormat Fields**6.2.5.1.10.1.1 NPixelFormat.Grayscale Field**

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

C#

```
public static readonly NPixelFormat Grayscale = new NPixelFormat(0x00301001);
```

6.2.5.1.10.1.2 NPixelFormat.Monochrome Field

Each pixel value is stored in 1 bit representing either black (value 0) or white (value 1).

C#

```
public static readonly NPixelFormat Monochrome = new NPixelFormat(0x00001001);
```

6.2.5.1.10.1.3 NPixelFormat.Rgb Field

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

C#

```
public static readonly NPixelFormat Rgb = new NPixelFormat(0x00303003);
```

6.2.5.1.10.2 NPixelFormat Methods

6.2.5.1.10.2.1 CalcRowLongSize Method

6.2.5.1.10.2.1.1 NPixelFormat.CalcRowLongSize Method (uint, uint)

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

C#

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

Parameters

Parameters	Description
uint bitCount	A number of bytes needed to store line of pixels.
uint length	A length of pixels.

Returns

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

6.2.5.1.10.2.1.2 NPixelFormat.CalcRowLongSize Method (uint, uint, uint)

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

C#

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

Parameters

Parameters	Description
uint bitCount	A number of bytes needed to store line of pixels.
uint length	A length of pixels.
uint alignment	Alignment.

Returns

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

6.2.5.1.10.2.2 CalcRowSize Method

6.2.5.1.10.2.2.1 NPixelFormat.CalcRowSize Method (uint, uint)

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

C#

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

Parameters

Parameters	Description
uint bitCount	A number of bytes needed to store line of pixels.
uint length	A length of pixels.

Returns

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

6.2.5.1.10.2.2 NPixelFormat.CalcRowSize Method (uint, uint, uint)

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

C#

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

Parameters

Parameters	Description
uint bitCount	A number of bytes needed to store line of pixels.
uint length	A length of pixels.
uint alignment	Alignment.

Returns

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

6.2.5.1.10.2.3 NPixelFormat.Equals Method

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

C#

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

Parameters

Parameters	Description
object obj	The Object to compare with the current Object.

Returns

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

6.2.5.1.10.2.4 NPixelFormat.GetHashCode Method

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

C#

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

Returns

A hash code for the current Object.

6.2.5.1.10.2.5 GetRowLongSize Method**6.2.5.1.10.2.5.1 NPixelFormat.GetRowLongSize Method (uint)**

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat (see page 870).

C#

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

Parameters

Parameters	Description
uint length	A length of pixels.

Returns

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

6.2.5.1.10.2.5.2 NPixelFormat.GetRowLongSize Method (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat (see page 870) and alignment.

C#

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

Parameters

Parameters	Description
uint length	A length of pixels.
uint alignment	Alignment.

Returns

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

6.2.5.1.10.2.6 GetRowSize Method**6.2.5.1.10.2.6.1 NPixelFormat.GetRowSize Method (uint)**

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat (see page 870).

C#

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

Parameters

Parameters	Description
uint length	A length of pixels.

Returns

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

6.2.5.1.10.2.6.2 NPixelFormat.GetRowSize Method (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat (see page 870) and alignment.

C#

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

Parameters

Parameters	Description
uint length	A length of pixels.
uint alignment	Alignment.

Returns

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

6.2.5.1.10.2.7 NPixelFormat.IsValid Method

Checks whether current NPixelFormat (see page 870) value is valid.

C#

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

Parameters

Parameters	Description
NPixelFormat value	The NPixelFormat (see page 870) object.

Returns

true if the object is valid NPixelFormat (see page 870), *false* if not.

6.2.5.1.10.2.8 NPixelFormat.ToString Method

Retrieves string representation of NPixelFormat (see page 870) object.

C#

```
public override string ToString();
```

Returns

String representation of NPixelFormat (see page 870) object.

6.2.5.1.10.3 NPixelFormat Operators**6.2.5.1.10.3.1 NPixelFormat.!= Operator**

Checks if two pixelFormats are not equal.

C#

```
public static bool operator !=(NPixelFormat pixelFormat1, NPixelFormat pixelFormat2);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat1	first pixelFormat to be compared.
NPixelFormat pixelFormat2	second pixelFormat to be compared.

Returns

True if pixelFormats are not equal, false if equal.

6.2.5.1.10.3.2 NPixelFormat.== Operator

Checks if two pixelFormats are equal.

C#

```
public static bool operator ==(NPixelFormat pixelFormat1, NPixelFormat pixelFormat2);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat1	first pixelFormat to be compared.
NPixelFormat pixelFormat2	second pixelFormat to be compared.

Returns

True if pixelFormats are equal, false otherwise.

6.2.5.1.10.3.3 NPixelFormat.NPixelFormat Operator

This conversion operator converts from a uint ([see page 876](#)) type to a NPixelFormat type

C#

```
public static explicit operator NPixelFormat(uint value);
```

Parameters

Parameters	Description
uint value	value to be converted

6.2.5.1.10.3.4 NPixelFormat.uint Operator

Gets value field of NPixelFormat ([see page 870](#)) object.

C#

```
public static explicit operator uint(NPixelFormat pixelFormat);
```

Parameters

Parameters	Description
NPixelFormat pixelFormat	NPixelFormat (see page 870) object.

6.2.5.1.10.4 NPixelFormat Properties

6.2.5.1.10.4.1 NPixelFormat.BitsPerPixel Property

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

C#

```
public uint BitsPerPixel;
```

Property value

A number of bits. sa NPixelFormat ([see page 870](#)).

6.2.5.1.11 NRgb Structure

Represents an RGB color.

C#

```
[StructLayout(LayoutKind.Sequential)]  
public struct NRgb {  
}
```

Methods

	Name	Description
	NRgb (see page 877)	Initializes a new instance of the NRgb structure.

NRgb Methods

	Name	Description
	ToString (see page 877)	Retrieves a string representation of NRgb object.

NRgb Properties

	Name	Description
	Blue (see page 877)	Gets the blue component value of this NRGB structure.
	Green (see page 877)	Gets the green component value of this NRGB structure.
	Red (see page 878)	Gets the red component value of this NRGB structure.

6.2.5.1.11.1 NRgb.NRgb Constructor

Initializes a new instance of the NRgb structure.

C#

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

Parameters

Parameters	Description
byte red	The blue component value of this NRGB.
byte green	The green component value of this NRGB.
byte blue	The red component value of this NRGB.

6.2.5.1.11.2 NRgb Methods

6.2.5.1.11.2.1 NRgb.ToString Method

Retrieves a string representation of NRgb ([see page 876](#)) object.

C#

```
public override string ToString();
```

Returns

String representation of NRgb ([see page 876](#)) object.

6.2.5.1.11.3 NRgb Properties

6.2.5.1.11.3.1 NRgb.Blue Property

Gets the blue component value of this NRGB structure.

C#

```
public byte Blue;
```

Property value

The blue component value of this NRGB.

6.2.5.1.11.3.2 NRgb.Green Property

Gets the green component value of this NRGB structure.

C#

```
public byte Green;
```

Property value

The green component value of this NRGB.

6.2.5.1.11.3.3 NRgb.Red Property

Gets the red component value of this NRGB structure.

C#

```
public byte Red;
```

Property value

The red component value of this NRGB.

6.2.5.1.12 NRgbImage Class

Provides functionality for managing 24-bit RGB images.

Class Hierarchy

C#

```
public sealed class NRgbImage : NImage;
```

Remarks

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

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
 	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
 	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

NImage Class

	Name	Description
	Clone (see page 835)	Creates NImage (see page 834) object from another NImage (see page 834) object.
 	Copy (see page 836)	Copies specified rectangular area of the NImage (see page 834) to other NImage (see page 834) object.
 	Create (see page 836)	Creates an image with specified pixel format, size, stride and resolution.
	Crop (see page 837)	Crops an image.
	FlipDiagonally (see page 838)	Flips image diagonally.
	FlipHorizontally (see page 838)	Flips image horizontally.
	FlipVertically (see page 838)	Flips image vertically.
 	FromBitmap (see page 838)	Creates NImage (see page 834) from Bitmap.
 	FromData (see page 838)	Creates NImage (see page 834) object from data with specified resolution.
 	FromFile (see page 843)	Creates NImage (see page 834) object from file.
 	FromHandle (see page 844)	Creates NImage (see page 834) object from specified handle.

	FromHBitmap (see page 844)	Creates NImage (see page 834) object from handle.
	FromImage (see page 844)	Creates NImage (see page 834) object from another NImage (see page 834) object.
	GetWrapper (see page 848)	Creates NImage (see page 834) object wrapper.
	RotateFlip (see page 854)	Rotates image using rotation type.
	Save (see page 854)	Saves NImage (see page 834) object to file.
	ToBitmap (see page 854)	Creates a Bitmap.
	ToHBitmap (see page 854)	Creates Windows HBITMAP.

NRgbImage Class

	Name	Description
	this (see page 879)	Gets or sets pixel value in NRGBImage.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

NImage Class

	Name	Description
	Height (see page 855)	Gets height of image from NImage (see page 834) object.
	HorzResolution (see page 855)	Gets horizontal resolution in pixels per inch of image.
	LongSize (see page 855)	Gets size of NImage (see page 834) object.
	LongStride (see page 855)	Gets stride of image from NImage (see page 834) object.
	PixelFormat (see page 856)	Gets NPixelFormat (see page 870) of NImage (see page 834) object.
	Pixels (see page 856)	Gets pointer to array of pixels from NImage (see page 834) object.
	Size (see page 856)	Gets size of NImage (see page 834) object.
	Stride (see page 856)	Gets stride of image from NImage (see page 834) object.
	VertResolution (see page 857)	Gets vertical resolution in pixels per inch of image.
	Width (see page 857)	Gets width of image from NImage (see page 834) object.

6.2.5.1.12.1 NRGBImage Methods**6.2.5.1.12.1.1 NRGBImage.this Indexer**

Gets or sets pixel value in NRGBImage ([see page 878](#)).

C#

```
public NRgb this[uint x, uint y];
```

Parameters

Parameters	Description
uint x	The X coordinate of a pixel
uint y	The Y coordinate of a pixel

Returns

Value of a pixel with specified coordinates.

6.2.5.1.13 Png Class

Provides functionality for loading and saving images in Png format.

Class Hierarchy

C#

```
public static class Png;
```

Png Fields

	Name	Description
	DefaultCompressionLevel (see page 880)	Specifies default Png compression level.

Png Methods

	Name	Description
	LoadImage (see page 880)	Creates NImage (see page 834) object from memory buffer.
	SaveImage (see page 881)	Saves NImage (see page 834) object to byte array in Png format.

6.2.5.1.13.1 Png Fields

6.2.5.1.13.1.1 Png.DefaultCompressionLevel Field

Specifies default Png ([see page 880](#)) compression level.

C#

```
public const int DefaultCompressionLevel = 6;
```

6.2.5.1.13.2 Png Methods

6.2.5.1.13.2.1 LoadImage Method

6.2.5.1.13.2.1.1 Png.LoadImage Method (IntPtr, int)

Creates NImage ([see page 834](#)) object from memory buffer.

C#

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

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer.
int bufferLength	Size of memory buffer.

Returns

A NImage ([see page 834](#)) object.

6.2.5.1.13.2.1.2 Png.LoadImage Method (Stream)

Creates NImage ([see page 834](#)) object from memory stream.

C#

```
public static NImage LoadImage(Stream stream);
```

Parameters

Parameters	Description
Stream stream	Memory stream which contains image data.

Returns

Created NImage (see page 834) object.

6.2.5.1.13.2.1.3 Png.LoadImage Method (byte[])

Creates NImage (see page 834) object from byte array.

C#

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

Parameters

Parameters	Description
byte[] buffer	A byte array.

Returns

A NImage (see page 834) object.

6.2.5.1.13.2.1.4 Png.LoadImage Method (string)

Creates NImage (see page 834) object from Png (see page 880) file.

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.

Returns

A NImage (see page 834) object.

6.2.5.1.13.2.2 SaveImage Method**6.2.5.1.13.2.2.1 Png.SaveImage Method (NImage)**

Saves NImage (see page 834) object to byte array in Png (see page 880) format.

C#

```
public static NBuffer SaveImage(NImage image);
```

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.

Returns

A byte array.

6.2.5.1.13.2.2.2 Png.SaveImage Method (NImage, Stream)

Saves NImage (see page 834) object to stream in Png (see page 880) format.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
Stream stream	The data stream used to save the image.

6.2.5.1.13.2.2.3 Png.SaveImage Method (NImage, int)

Saves NImage (see page 834) object to byte array in Png (see page 880) format with specified bit rate.

C#

```
public static NBuffer SaveImage(NImage image, int compressionLevel);
```

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
int compressionLevel	Specifies compression level of Png (see page 880) image.

Returns

A byte array.

6.2.5.1.13.2.2.4 Png.SaveImage Method (NImage, int, Stream)

Saves NImage (see page 834) object to stream in Png (see page 880) format with specified bit rate.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
int compressionLevel	Specifies compression level of Png (see page 880) image.
Stream stream	The data stream used to save the image.

6.2.5.1.13.2.2.5 Png.SaveImage Method (NImage, int, string)

Saves NImage (see page 834) object to file in Png (see page 880) format with specified bit rate.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
int compressionLevel	Specifies compression level of Png (see page 880) image.
string fileName	A string that contains the name of the file.

6.2.5.1.13.2.2.6 Png.SaveImage Method (NImage, string)

NImage (see page 834) object to file in Png (see page 880) format.

C#

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

Parameters

Parameters	Description
NImage image	A NImage (see page 834) object.
string fileName	A string that contains the name of the file.

6.2.5.1.14 Tiff Class

Provides functionality for loading and saving images in TIFF format.

Class Hierarchy**C#**

```
public static class Tiff;
```

Tiff Methods

	Name	Description
	LoadImage (see page 883)	Creates NImage (see page 834) object from memory buffer.

6.2.5.1.14.1 Tiff Methods

6.2.5.1.14.1.1 LoadImage Method

6.2.5.1.14.1.1.1 Tiff.LoadImage Method (IntPtr, int)

Creates NImage (see page 834) object from memory buffer.

C#

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

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer.
int bufferLength	Size of memory buffer.

Returns

A NImage (see page 834) object.

6.2.5.1.14.1.1.2 Tiff.LoadImage Method (Stream)

Creates NImage (see page 834) object from memory buffer.

C#

```
public static NImage LoadImage(Stream stream);
```

Parameters

Parameters	Description
Stream stream	Memory stream which contains Tiff (see page 883) image data.

Returns

Loaded NImage (see page 834) object.

6.2.5.1.14.1.1.3 Tiff.LoadImage Method (byte[])

Creates NImage (see page 834) object from byte array.

C#

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

Parameters

Parameters	Description
byte[] buffer	A byte array. return A NImage (see page 834) object.

Returns

A NImage (see page 834) object.

6.2.5.1.14.1.1.4 Tiff.LoadImage Method (string)

Creates NImage (see page 834) object from TIFF file.

C#

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

Parameters

Parameters	Description
string fileName	A string that contains the name of the file.

Returns

A NImage (see page 834) object.

6.2.5.2 Structs, Records, Enums

6.2.5.2.1 Neurotec.Images.BmpRecording Enumeration

Enumerates different bitmap image pixels arrangement types.

C#

```
internal enum BmpRecording {  
    TopDown = 0,  
    BottomUp = 1  
}
```

Members

Members	Description
TopDown = 0	Pixels are stored in the way starting in the upper left corner, going from left to right, and then row by row from the top to the bottom of the image.
BottomUp = 1	Pixels are stored in the way starting in the lower left corner, going from left to right, and then row by row from the bottom to the top of the image.

6.2.5.2.2 Neurotec.Images.BmpSaveMode Enumeration

Enumerates different bitmap image data storage types.

C#

```
internal enum BmpSaveMode {  
}
```

6.2.5.2.3 Neurotec.Images.NImageRotateFlipType Enumeration

Enumerates NImage (see page 834) rotation and flip values.

C#

```
[Flags]
public enum NImageRotateFlipType {
    RotateNone = 0,
    Rotate90 = 1,
    Rotate180 = 2,
    Rotate270 = 3,
    FlipNone = 0,
    FlipX = 4,
    FlipY = 8,
    FlipXY = FlipX|FlipY,
    None = RotateNone|FlipNone
}
```

Members

Members	Description
RotateNone = 0	Indicates that image should not be rotated.
Rotate90 = 1	Rotates image by 90 degrees.
Rotate180 = 2	Rotates image by 180 degrees.
Rotate270 = 3	Rotates image by 270 degrees.
FlipNone = 0	No flip.
FlipX = 4	Flips the image's orientation of the X coordinates.
FlipY = 8	Flips the image's orientation of the Y coordinates.
FlipXY = FlipX FlipY	Flips the image's orientation of the X or Y coordinates.
None = RotateNone FlipNone	No flip or rotation.

6.2.6 Neurotec.IO Namespace

Classes under this namespace provides infrastructure for Neurotechnology components.

Classes

	Name	Description
	NBuffer (see page 886)	Provides methods for manipulating arrays of primitive types.
	NMemoryStream (see page 887)	Provides access to unmanaged blocks of memory from managed code.
	NStream (see page 889)	This class supports internal Neurotechnology libraries infrastructure and should not be used directly in your code.

Module

.NET Reference (see page 606)

Structs, Records, Enums

	Name	Description
	NByteOrder (see page 895)	Specifies byte order.

6.2.6.1 Classes

6.2.6.1.1 NBuffer Class

Provides methods for manipulating arrays of primitive types.

Class Hierarchy

C#

```
public class NBuffer : NDisposable;
```

Methods

NBuffer Class

	Name	Description
	NBuffer (see page 886)	Initializes a new instance of the NBuffer class.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NBuffer Class

	Name	Description
	ToArray (see page 886)	Copies the elements of the NBuffer to a new byte array.
	WriteTo (see page 887)	Writes the contents of NBuffer to memory stream.

NBuffer Properties

NBuffer Class

	Name	Description
	Length (see page 887)	Gets the length of NBuffer.
	LongLength (see page 887)	Gets a 64-bit integer that represents the length of NBuffer.
	Ptr (see page 887)	Gets a representation of NBuffer pointer.
	UIntPtrLength (see page 887)	Gets UIntPtr that represents the length of NBuffer.

6.2.6.1.1.1 NBuffer.NBuffer Constructor

Initializes a new instance of the NBuffer class.

C#

```
public NBuffer(IntPtr ptr, int length);
```

Parameters

Parameters	Description
IntPtr ptr	A pointer to buffer.
int length	Length (see page 887) of a buffer.

6.2.6.1.1.2 NBuffer Methods

6.2.6.1.1.2.1 NBuffer.ToArray Method

Copies the elements of the NBuffer ([see page 886](#)) to a new byte array.

C#

```
public byte[] ToArray();
```

Returns

A byte array containing copies of the elements of the NBuffer (see page 886).

6.2.6.1.1.2.2 NBuffer.WriteTo Method

Writes the contents of NBuffer (see page 886) to memory stream.

C#

```
public void WriteTo(Stream stream);
```

Parameters

Parameters	Description
Stream stream	The stream to write the contents of NBuffer (see page 886) to.

6.2.6.1.1.3 NBuffer Properties

6.2.6.1.1.3.1 NBuffer.Length Property

Gets the length of NBuffer (see page 886).

C#

```
public int Length;
```

Property value

The length of NBuffer (see page 886).

6.2.6.1.1.3.2 NBuffer.LongLength Property

Gets a 64-bit integer that represents the length of NBuffer (see page 886).

C#

```
public long LongLength;
```

6.2.6.1.1.3.3 NBuffer.Ptr Property

Gets a representation of NBuffer (see page 886) pointer.

C#

```
public IntPtr Ptr;
```

6.2.6.1.1.3.4 NBuffer.UIntPtrLength Property

Gets UIntPtr that represents the length of NBuffer (see page 886).

C#

```
public UIntPtr UIntPtrLength;
```

6.2.6.1.2 NMemoryStream Class

Provides access to unmanaged blocks of memory from managed code.

Class Hierarchy

C#

```
public sealed class NMemoryStream : NStream;
```


Methods**NMemoryStream Class**

	Name	Description
	NMemoryStream (see page 888)	Initializes a new, empty instance of the NMemoryStream class.

NStream Methods

	Name	Description
	Close (see page 890)	Closes current stream.
	Flush (see page 890)	Clears all buffers for this stream and causes any buffered data to be written to the underlying device.
 	FromHandle (see page 890)	Gets NStream (see page 889) from it's handle.
 	FromStream (see page 891)	Creates new NStream (see page 889) from Stream object.
	Read (see page 891)	Reads a sequence of bytes from the current stream.
	ReadByte (see page 892)	Reads a byte from the stream.
	Seek (see page 892)	Sets the position within the current stream.
	SetLength (see page 893)	Sets the length of the current stream.
	Write (see page 893)	Writes a sequence of bytes to the current stream.
	WriteByte (see page 894)	Writes a byte to the current position in the stream.

NMemoryStream Class

	Name	Description
	ToArray (see page 889)	Saves stream to byte array.
	WriteTo (see page 889)	Writes NMemoryStream to Stream.

NStream Properties

	Name	Description
	CanRead (see page 894)	Gets a value indicating whether the current stream supports reading.
	CanSeek (see page 894)	Gets a value indicating whether the current stream supports seeking.
	CanWrite (see page 894)	Gets a value indicating whether the current stream supports writing.
	Handle (see page 894)	Gets handle to unmanaged NStream (see page 889) object.
	Length (see page 894)	Gets size of a stream in bytes.
	Position (see page 894)	Gets number of bytes from the start of the data to the current position in a stream.

NMemoryStream Class

	Name	Description
	Capacity (see page 889)	Gets the stream size of memory assigned to a stream (capacity).

6.2.6.1.2.1 NMemoryStream Constructor**6.2.6.1.2.1.1 NMemoryStream.NMemoryStream Constructor ()**

Initializes a new, empty instance of the NMemoryStream class.

C#

```
public NMemoryStream();
```

6.2.6.1.2.1.2 NMemoryStream.NMemoryStream Constructor (long)

Initializes a new instance of the NMemoryStream class using the specified memory length.

C#

```
public NMemoryStream(long capacity);
```

Parameters

Parameters	Description
long capacity	The length of the memory to use.

6.2.6.1.2.1.3 NMemoryStream.NMemoryStream Constructor (long, long)

Initializes a new instance of the NMemoryStream class using the specified memory length, total amount of memory.

C#

```
public NMemoryStream(long capacity, long maxCapacity);
```

Parameters

Parameters	Description
long capacity	The length of the memory to use.
long maxCapacity	The max amount of memory assigned to the stream.

6.2.6.1.2.2 NMemoryStream Methods

6.2.6.1.2.2.1 NMemoryStream.ToArray Method

Saves stream to byte array.

C#

```
public byte[] ToArray();
```

Returns

Byte array.

6.2.6.1.2.2.2 NMemoryStream.WriteTo Method

Writes NMemoryStream (see page 887) to Stream.

C#

```
public void WriteTo(Stream stream);
```

Parameters

Parameters	Description
Stream stream	Stream object.

6.2.6.1.2.3 NMemoryStream Properties

6.2.6.1.2.3.1 NMemoryStream.Capacity Property

Gets the stream size of memory assigned to a stream (capacity).

C#

```
public long Capacity;
```

6.2.6.1.3 NStream Class

This class supports internal Neurotechnology libraries infrastructure and should not be used directly in your code.

Class Hierarchy

C#

```
public class NStream : Stream;
```

NStream Methods

	Name	Description
	Close (see page 890)	Closes current stream.
	Flush (see page 890)	Clears all buffers for this stream and causes any buffered data to be written to the underlying device.
 	FromHandle (see page 890)	Gets NStream from it's handle.
 	FromStream (see page 891)	Creates new NStream from Stream object.
	Read (see page 891)	Reads a sequence of bytes from the current stream.
	ReadByte (see page 892)	Reads a byte from the stream.
	Seek (see page 892)	Sets the position within the current stream.
	SetLength (see page 893)	Sets the length of the current stream.
	Write (see page 893)	Writes a sequence of bytes to the current stream.
	WriteByte (see page 894)	Writes a byte to the current position in the stream.

NStream Properties

	Name	Description
	CanRead (see page 894)	Gets a value indicating whether the current stream supports reading.
	CanSeek (see page 894)	Gets a value indicating whether the current stream supports seeking.
	CanWrite (see page 894)	Gets a value indicating whether the current stream supports writing.
	Handle (see page 894)	Gets handle to unmanaged NStream object.
	Length (see page 894)	Gets size of a stream in bytes.
	Position (see page 894)	Gets number of bytes from the start of the data to the current position in a stream.

6.2.6.1.3.1 NStream Methods**6.2.6.1.3.1.1 NStream.Close Method**

Closes current stream.

C#

```
public override void Close();
```

6.2.6.1.3.1.2 NStream.Flush Method

Clears all buffers for this stream and causes any buffered data to be written to the underlying device.

C#

```
public override void Flush();
```

6.2.6.1.3.1.3 FromHandle Method**6.2.6.1.3.1.3.1 NStream.FromHandle Method (IntPtr)**

Gets NStream ([see page 889](#)) from it's handle.

C#

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

Parameters

Parameters	Description
IntPtr handle	Handle (see page 894) to unmanaged NStream (see page 889) object.

Returns

NStream (see page 889) object.

6.2.6.1.3.1.3.2 NStream.FromHandle Method (IntPtr, bool)

Gets NStream (see page 889) from it's handle.

C#

```
public static NStream FromHandle(IntPtr handle, bool ownsHandle);
```

Parameters

Parameters	Description
IntPtr handle	Handle (see page 894) to unmanaged NStream (see page 889) object.
bool ownsHandle	If set to true, on destruction of the NStream (see page 889) the NStream (see page 889) pointed to by the handle will also be destroyed.

Returns

NStream (see page 889) object.

6.2.6.1.3.1.4 NStream.FromStream Method

Creates new NStream (see page 889) from Stream object.

C#

```
public static NStream FromStream(Stream stream, bool ownsStream);
```

Parameters

Parameters	Description
Stream stream	Memory stream.
bool ownsStream	If set to true, on destruction of the NStream (see page 889) the original Stream will also be destroyed.

Returns

NStream (see page 889) object.

6.2.6.1.3.1.5 Read Method**6.2.6.1.3.1.5.1 NStream.Read Method (IntPtr, int)**

Reads a sequence of bytes from the current stream.

C#

```
public int Read(IntPtr pBuffer, int count);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer.
int count	Number of bytes to be read.

Returns

Integer type variable with number of bytes successfully read.

6.2.6.1.3.1.5.2 NStream.Read Method (IntPtr, long)

Reads a sequence of bytes from the current stream.

C#

```
public long Read(IntPtr pBuffer, long count);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer.
long count	Number of bytes to be read.

Returns

Long type variable with number of bytes successfully read.

6.2.6.1.3.1.5.3 NStream.Read Method (byte[], int, int)

Reads a sequence of bytes from the current stream.

C#

```
public override int Read(byte[] buffer, int offset, int count);
```

Parameters

Parameters	Description
byte[] buffer	An array of bytes. When this method returns, the buffer contains the specified byte array with the values between offset and (offset + count - 1) replaced by the bytes read from the current source.
int offset	The zero-based byte offset in buffer at which to begin storing the data read from the current stream.
int count	Number of bytes to be read from the current stream.

Returns

Integer type variable with number of bytes successfully read.

6.2.6.1.3.1.6 NStream.ReadByte Method

Reads a byte from the stream.

C#

```
public override int ReadByte();
```

Returns

Value of byte at current position in a stream.

6.2.6.1.3.1.7 NStream.Seek Method

Sets the position within the current stream.

C#

```
public override long Seek(long offset, SeekOrigin origin);
```

Parameters

Parameters	Description
long offset	A byte offset relative to the origin parameter.
SeekOrigin origin	A value indicating the reference point used to obtain the new position.

Returns

The new position within the current stream.

6.2.6.1.3.1.8 NStream.SetLength Method

Sets the length of the current stream.

C#

```
public override void SetLength(long value);
```

Parameters

Parameters	Description
long value	New desired length of the current stream in bytes.

6.2.6.1.3.1.9 Write Method**6.2.6.1.3.1.9.1 NStream.Write Method (IntPtr, int)**

Writes a sequence of bytes to the current stream.

C#

```
public void Write(IntPtr pBuffer, int count);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer.
int count	Number of bytes to be written.

6.2.6.1.3.1.9.2 NStream.Write Method (IntPtr, long)

Writes a sequence of bytes to the current stream.

C#

```
public void Write(IntPtr pBuffer, long count);
```

Parameters

Parameters	Description
IntPtr pBuffer	Memory buffer.
long count	Number of bytes to be written.

6.2.6.1.3.1.9.3 NStream.Write Method (byte[], int, int)

Writes a sequence of bytes to the current stream.

C#

```
public override void Write(byte[] buffer, int offset, int count);
```

Parameters

Parameters	Description
byte[] buffer	An array of bytes.

int offset	The zero-based byte offset in buffer at which to begin copying bytes to the current stream.
int count	Number of bytes to be written to the current stream.

6.2.6.1.3.1.10 NStream.WriteByte Method

Writes a byte to the current position in the stream.

C#

```
public override void WriteByte(byte value);
```

Parameters

Parameters	Description
byte value	The byte to write to the stream.

6.2.6.1.3.2 NStream Properties

6.2.6.1.3.2.1 NStream.CanRead Property

Gets a value indicating whether the current stream supports reading.

C#

```
public override bool CanRead;
```

6.2.6.1.3.2.2 NStream.CanSeek Property

Gets a value indicating whether the current stream supports seeking.

C#

```
public override bool CanSeek;
```

6.2.6.1.3.2.3 NStream.CanWrite Property

Gets a value indicating whether the current stream supports writing.

C#

```
public override bool CanWrite;
```

6.2.6.1.3.2.4 NStream.Handle Property

Gets handle to unmanaged NStream (see page 889) object.

C#

```
public IntPtr Handle;
```

6.2.6.1.3.2.5 NStream.Length Property

Gets size of a stream in bytes.

C#

```
public override long Length;
```

6.2.6.1.3.2.6 NStream.Position Property

Gets number of bytes from the start of the data to the current position in a stream.

C#

```
public override long Position;
```

6.2.6.2 Structs, Records, Enums

6.2.6.2.1 Neurotec.IO.NByteOrder Enumeration

Specifies byte order.

```
C#
public enum NByteOrder {
    LittleEndian = 0,
    BigEndian = 1
}
```

Members

Members	Description
LittleEndian = 0	Little endian byte order is used.
BigEndian = 1	Big endian byte order is used.

6.2.7 Neurotec.Licensing Namespace

Provides functionality for getting, releasing licenses.

Classes

	Name	Description
	NLicense (see page 895)	Provides functionality for getting, releasing licenses.
	NLicenseInfo (see page 897)	Provides information about NLicensing (see page 260) library.
	NLicensing (see page 898)	Provides methods for retrieving information about the NLicensing (see page 260) library.

Module

.NET Reference ([see page 606](#))

6.2.7.1 Classes

6.2.7.1.1 NLicense Class

Provides functionality for getting, releasing licenses.

Class Hierarchy

```
C#
public static class NLicense;
```

NLicense Methods

	Name	Description
	GetInfo (see page 896)	Retrieves license information of specified product.
	IsComponentActivated (see page 896)	Checks if component specified by name is activated.
	Obtain (see page 896)	Obtains license.
	Release (see page 897)	Releases license.

6.2.7.1.1.1 NLicense Methods

6.2.7.1.1.1.1 NLicense.GetInfo Method

Retrieves license information of specified product.

C#

```
public static NLicenseInfo GetInfo(string product);
```

Parameters

Parameters	Description
string product	String that contains name of a product to retrieve license information.

Returns

NLicenseInfo (see page 897) object that contains properties for retrieving license information.

6.2.7.1.1.1.2 NLicense.IsComponentActivated Method

Checks if component specified by name is activated.

C#

```
public static bool IsComponentActivated(string name);
```

Parameters

Parameters	Description
string name	String which contains component name.

Returns

Bool value indicating if the specified component was activated.

6.2.7.1.1.1.3 Obtain Method

6.2.7.1.1.1.3.1 NLicense.Obtain Method (string, int, string)

Obtains license.

C#

```
public static bool Obtain(string address, int port, string product);
```

Parameters

Parameters	Description
string address	Server address where license manager is installed as a server. "/local" is used for this parameter when license is tied with the computer using serial number.
int port	License manager server port.
string product	String which contains license name.

Returns

Returns true if license available; otherwise, false.

6.2.7.1.1.1.3.2 NLicense.Obtain Method (string, string, string)

Obtains license.

C#

```
public static bool Obtain(string address, string port, string product);
```

Parameters

Parameters	Description
string address	Server address where license manager is installed as a server. "/local" is used for this parameter when license is tied with the computer using serial number.
string port	License manager server port.
string product	String which contains license name.

Returns

Returns true if license available; otherwise, false.

6.2.7.1.1.1.4 NLicense.Release Method

Releases license.

C#

```
public static void Release(string product);
```

Parameters

Parameters	Description
string product	Name of a product to release its license.

Returns

Returns true if license available; otherwise, false.

6.2.7.1.2 NLicenseInfo Class

Provides information about NLicensing ([see page 260](#)) library.

Class Hierarchy**C#**

```
[StructLayout(LayoutKind.Sequential)]  
public sealed class NLicenseInfo;
```

NLicenseInfo Properties

	Name	Description
	DistributorId (see page 897)	Gets distributor id.
	IsObtained (see page 898)	Gets license status.
	SerialNumber (see page 898)	Gets serial number of a license.

6.2.7.1.2.1 NLicenseInfo Properties**6.2.7.1.2.1.1 NLicenseInfo.DistributorId Property**

Gets distributor id.

C#

```
public int DistributorId;
```

Property value

Distributor id.

6.2.7.1.2.1.2 NLicenseInfo.IsObtained Property

Gets license status.

```
C#
public bool IsObtained;
```

Property value

True if license is obtained; otherwise, false.

6.2.7.1.2.1.3 NLicenseInfo.SerialNumber Property

Gets serial number of a license.

```
C#
public int SerialNumber;
```

Returns

Serial number.

6.2.7.1.3 NLicensing Class

Provides methods for retrieving information about the NLicensing (see page 260) library.

Class Hierarchy

```
C#
public static class NLicensing;
```

NLicensing Fields

	Name	Description
	DllName (see page 898)	Name of DLL containing unmanaged part of this class.

NLicensing Methods

	Name	Description
	GetInfo (see page 898)	Gets information about the library.

6.2.7.1.3.1 NLicensing Fields

6.2.7.1.3.1.1 NLicensing.DllName Field

Name of DLL containing unmanaged part of this class.

```
C#
public const string DllName = "NLicensing";
```

6.2.7.1.3.2 NLicensing Methods

6.2.7.1.3.2.1 NLicensing.GetInfo Method

Gets information about the library.

```
C#
public static NLibraryInfo GetInfo();
```

Returns

Returns NLibraryInfo (see page 619) structure with library information.

6.2.8 Neurotec.SmartCards Namespace

Contains classes for working with smartcards.

Remarks

Neurotec.SmartCards library provides functionality for working with ISO/IEC 7816-4:2005 compliant SIMPLE-TLV and interindustry BER-TLV data objects. It also contains classes that encapsulate Win32 smartcard API and enable transmission of standardized ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005 compliant APDU commands (C-APDUs) to the card as well as retrieval of corresponding APDU responses (R-APDUs).

Classes

	Name	Description
	ApduClass (see page 899)	Represents APDU command class.
	ApduInstruction (see page 904)	Represents APDU command instruction.
	ApduStatus (see page 916)	Represents APDU response status.
	BerTag (see page 941)	Represents BER-TLV tag.
	BerTlv (see page 952)	Provides functionality for creation, validation, retrieval and update of BER-TLV data objects that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.
	ConstructedBerTlv (see page 961)	Provides functionality for creation, validation, retrieval and update of constructed BER-TLV data objects that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.
	PrimitiveBerTlv (see page 967)	Provides functionality for creation, validation, retrieval and update of primitive BER-TLV data objects that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.
	SCardTags (see page 972)	Defines BER-TLV tags that can be used during communication with a smartcard.
	SimpleTag (see page 991)	Represents SIMPLE-TLV tag.
	SimpleTlv (see page 993)	Provides functionality for creation, validation, retrieval and update of SIMPLE-TLV data objects that are fully compliant with SIMPLE-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Module

.NET Reference ([see page 606](#))

Structs, Records, Enums

	Name	Description
	ApduClassSecureMessaging (see page 1001)	Specifies secure messaging applied to the APDU command-response pairs according to ISO/IEC 7816-4:2005.
	BerTagClass (see page 1001)	Specifies a class of a BER-TLV data object.
	BerTagEncoding (see page 1002)	Specifies an encoding of a BER-TLV data object.

6.2.8.1 Classes

6.2.8.1.1 ApduClass Structure

Represents APDU command class.

C#

[Serializable]

```
public struct AduClass {
}
```

Methods

	Name	Description
	AduClass (see page 900)	Initializes a new instance of the AduClass .

AduClass Methods

	Name	Description
	Equals (see page 901)	Overloaded. Determines whether two Object instances are equal.
	GetHashCode (see page 901)	Serves as a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.
	ToString (see page 901)	Returns a String that represents the current Object.

AduClass Operators

	Name	Description
	!= (see page 901)	Gets a value indicating whether the AduClass is not identical to a second AduClass.
	== (see page 902)	Gets a value indicating whether the AduClass is identical to a second AduClass.
	byte (see page 902)	Converts APDU class to byte.

AduClass Properties

	Name	Description
	ChannelNumber (see page 902)	Represents APDU command class.
	IsInterindustry (see page 902)	Represents APDU command class.
	IsInterindustryFirst (see page 903)	Represents APDU command class.
	IsInterindustryFurther (see page 903)	Represents APDU command class.
	IsInterindustryReserved (see page 903)	Represents APDU command class.
	IsLastOrTheOnly (see page 903)	Represents APDU command class.
	IsProprietary (see page 903)	Represents APDU command class.
	IsValid (see page 904)	Represents APDU command class.
	SecureMessaging (see page 904)	Represents APDU command class.

6.2.8.1.1.1 AduClass Constructor

6.2.8.1.1.1.1 AduClass.AduClass Constructor (bool, AduClassSecureMessaging, int)

Initializes a new instance of the AduClass .

C#

```
public AduClass(bool isLastOrTheOnly, AduClassSecureMessaging secureMessaging, int
channelNumber);
```

Parameters

Parameters	Description
bool isLastOrTheOnly	Boolean parameter specifying whether the APDU command is the last or only command of a chain.
AduClassSecureMessaging secureMessaging	The identifier of a secure messaging mechanism applied to the APDU command.

int channelNumber	Number of a logical channel to pass the APDU command through.

6.2.8.1.1.1.2 AduClass.AduClass Constructor (byte)

Initializes a new instance of the AduClass .

C#

```
public AduClass(byte value);
```

Parameters

Parameters	Description
byte value	Class byte (see page 902) (CLA) of the APDU command header.

6.2.8.1.1.2 AduClass Methods

6.2.8.1.1.2.1 AduClass.Equals Method

Overloaded. Determines whether two Object instances are equal.

C#

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

Parameters

Parameters	Description
object obj	The object to compare with the current object.

Returns

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

6.2.8.1.1.2.2 AduClass.GetHashCode Method

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

C#

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

Returns

A hash code for the current Object.

6.2.8.1.1.2.3 AduClass.ToString Method

Returns a String that represents the current Object.

C#

```
public override string ToString();
```

Returns

Returns a String that represents the current Object.

6.2.8.1.1.3 AduClass Operators

6.2.8.1.1.3.1 AduClass.!= Operator

Gets a value indicating whether the AduClass (see page 899) is not identical to a second AduClass (see page 899).

C#

```
public static bool operator !=(ApuClass v1, ApduClass v2);
```

Parameters

Parameters	Description
ApduClass v1	ApduClass (see page 899) object.
ApduClass v2	ApduClass (see page 899) object.

Returns

true if its two operands refer to different objects; otherwise, false.

6.2.8.1.1.3.2 ApduClass.== Operator

Gets a value indicating whether the ApduClass (see page 899) is identical to a second ApduClass (see page 899).

C#

```
public static bool operator ==(ApuClass v1, ApduClass v2);
```

Parameters

Parameters	Description
ApduClass v1	ApduClass (see page 899) object.
ApduClass v2	ApduClass (see page 899) object.

Returns

true if the ApduClass (see page 899) is identical to a second ApduClass (see page 899); otherwise, false.

6.2.8.1.1.3.3 ApduClass.byte Operator

Converts APDU class to byte.

C#

```
public static explicit operator byte(ApuClass value);
```

Parameters

Parameters	Description
ApduClass value	ApduClass (see page 899) object.

Returns

ApduClass (see page 899) converted to byte.

6.2.8.1.1.4 ApduClass Properties**6.2.8.1.1.4.1 ApduClass.ChannelNumber Property**

Represents APDU command class.

C#

```
public int ChannelNumber;
```

Property value

Number of a logical channel to pass the APDU command through.

6.2.8.1.1.4.2 ApduClass.IsInterindustry Property

Represents APDU command class.

C#

```
public bool IsInterindustry;
```

Property value

APDU command class byte (see page 902) (CLA) value.

6.2.8.1.1.4.3 ApduClass.IsInterindustryFirst Property

Represents APDU command class.

C#

```
public bool IsInterindustryFirst;
```

Property value

APDU command class byte (see page 902) (CLA) value.

6.2.8.1.1.4.4 ApduClass.IsInterindustryFurther Property

Represents APDU command class.

C#

```
public bool IsInterindustryFurther;
```

Property value

APDU command class byte (see page 902) (CLA) value.

6.2.8.1.1.4.5 ApduClass.IsInterindustryReserved Property

Represents APDU command class.

C#

```
public bool IsInterindustryReserved;
```

Property value

APDU command class byte (see page 902) (CLA) value.

6.2.8.1.1.4.6 ApduClass.IsLastOrTheOnly Property

Represents APDU command class.

C#

```
public bool IsLastOrTheOnly;
```

Property value

Boolean parameter specifying whether the APDU command is the last or only command of a chain.

6.2.8.1.1.4.7 ApduClass.IsProprietary Property

Represents APDU command class.

C#

```
public bool IsProprietary;
```

Property value

APDU command class byte (see page 902) (CLA) value.

6.2.8.1.1.4.8 ApcduClass.IsValid Property

Represents APDU command class.

C#

```
public bool IsValid;
```

Property value

APDU command class byte (see page 902) (CLA) value.

6.2.8.1.1.4.9 ApcduClass.SecureMessaging Property

Represents APDU command class.

C#

```
public ApcduClassSecureMessaging SecureMessaging;
```

Property value

The identifier of a secure messaging mechanism applied to the APDU command.

6.2.8.1.2 ApcduInstruction Structure

Represents APDU command instruction.

C#

```
[Serializable]
public struct ApcduInstruction {
    public static readonly ApcduInstruction DeactivateFile = (ApcduInstruction)0x04;
    public static readonly ApcduInstruction EraseRecord = (ApcduInstruction)0x0C;
    public static readonly ApcduInstruction EraseBinary = (ApcduInstruction)0x0E;
    public static readonly ApcduInstruction EraseBinaryBerTlv = (ApcduInstruction)0x0F;
    public static readonly ApcduInstruction PerformScqlOperation = (ApcduInstruction)0x10;
    public static readonly ApcduInstruction PerformTransactionOperation = (ApcduInstruction)0x12;
    public static readonly ApcduInstruction PerformUserOperation = (ApcduInstruction)0x14;
    public static readonly ApcduInstruction Verify = (ApcduInstruction)0x20;
    public static readonly ApcduInstruction VerifyBerTlv = (ApcduInstruction)0x21;
    public static readonly ApcduInstruction ManageSecurityEnvironment = (ApcduInstruction)0x22;
    public static readonly ApcduInstruction ChangeReferenceData = (ApcduInstruction)0x24;
    public static readonly ApcduInstruction DisableVerificationRequirement = (ApcduInstruction)0x26;
    public static readonly ApcduInstruction EnableVerificationRequirement = (ApcduInstruction)0x28;
    public static readonly ApcduInstruction PerformSecurityOperation = (ApcduInstruction)0x2A;
    public static readonly ApcduInstruction ResetRetryCounter = (ApcduInstruction)0x2C;
    public static readonly ApcduInstruction ActivateFile = (ApcduInstruction)0x44;
    public static readonly ApcduInstruction GenerateAsymmetricKeyPair = (ApcduInstruction)0x46;
    public static readonly ApcduInstruction ManageChannel = (ApcduInstruction)0x70;
    public static readonly ApcduInstruction ExternalAuthenticate = (ApcduInstruction)0x82;
    public static readonly ApcduInstruction GetChallenge = (ApcduInstruction)0x84;
    public static readonly ApcduInstruction GeneralAuthenticate = (ApcduInstruction)0x86;
    public static readonly ApcduInstruction GeneralAuthenticateBerTlv = (ApcduInstruction)0x87;
    public static readonly ApcduInstruction InternalAuthenticate = (ApcduInstruction)0x88;
    public static readonly ApcduInstruction SearchBinary = (ApcduInstruction)0xA0;
    public static readonly ApcduInstruction SearchBinaryBerTlv = (ApcduInstruction)0xA1;
    public static readonly ApcduInstruction SearchRecord = (ApcduInstruction)0xA2;
    public static readonly ApcduInstruction Select = (ApcduInstruction)0xA4;
    public static readonly ApcduInstruction ReadBinary = (ApcduInstruction)0xB0;
    public static readonly ApcduInstruction ReadBinaryBerTlv = (ApcduInstruction)0xB1;
    public static readonly ApcduInstruction ReadRecord = (ApcduInstruction)0xB2;
    public static readonly ApcduInstruction ReadRecordBerTlv = (ApcduInstruction)0xB3;
    public static readonly ApcduInstruction GetResponse = (ApcduInstruction)0xC0;
    public static readonly ApcduInstruction Envelope = (ApcduInstruction)0xC2;
```

```

public static readonly AduInstruction EnvelopeBerTlv = (AduInstruction)0xC3;
public static readonly AduInstruction GetData = (AduInstruction)0xCA;
public static readonly AduInstruction GetDataBerTlv = (AduInstruction)0xCB;
public static readonly AduInstruction WriteBinary = (AduInstruction)0xD0;
public static readonly AduInstruction WriteBinaryBerTlv = (AduInstruction)0xD1;
public static readonly AduInstruction WriteRecord = (AduInstruction)0xD2;
public static readonly AduInstruction UpdateBinary = (AduInstruction)0xD6;
public static readonly AduInstruction UpdateBinaryBerTlv = (AduInstruction)0xD7;
public static readonly AduInstruction PutData = (AduInstruction)0xDA;
public static readonly AduInstruction PutDataBerTlv = (AduInstruction)0xDB;
public static readonly AduInstruction UpdateRecord = (AduInstruction)0xDC;
public static readonly AduInstruction UpdateRecordBerTlv = (AduInstruction)0xDD;
public static readonly AduInstruction CreateFile = (AduInstruction)0xE0;
public static readonly AduInstruction AppendRecord = (AduInstruction)0xE2;
public static readonly AduInstruction DeleteFile = (AduInstruction)0xE4;
public static readonly AduInstruction TerminateDF = (AduInstruction)0xE6;
public static readonly AduInstruction TerminateEF = (AduInstruction)0xE8;
public static readonly AduInstruction TerminateCardUsage = (AduInstruction)0xFE;
}

```

ApduInstruction Fields

	Name	Description
	ActivateFile (see page 908)	ApduInstruction for ACTIVATE FILE command as specified in ISO/IEC 7816-4:2005.
	AppendRecord (see page 908)	ApduInstruction for APPEND RECORD command as specified in ISO/IEC 7816-4:2005.
	ChangeReferenceData (see page 908)	ApduInstruction for CHANGE REFERENCE DATA command as specified in ISO/IEC 7816-4:2005.
	CreateFile (see page 908)	ApduInstruction for CREATE FILE command as specified in ISO/IEC 7816-4:2005.
	DeactivateFile (see page 908)	ApduInstruction for DEACTIVATE FILE command as specified in ISO/IEC 7816-4:2005.
	DeleteFile (see page 908)	ApduInstruction for DELETE FILE command as specified in ISO/IEC 7816-4:2005.
	DisableVerificationRequirement (see page 908)	ApduInstruction for DISABLE VERIFICATION REQUIREMENT command as specified in ISO/IEC 7816-4:2005.
	EnableVerificationRequirement (see page 909)	ApduInstruction for ENABLE VERIFICATION REQUIREMENT command as specified in ISO/IEC 7816-4:2005.
	Envelope (see page 909)	ApduInstruction for ENVELOPE command as specified in ISO/IEC 7816-4:2005.
	EnvelopeBerTlv (see page 909)	ApduInstruction for ENVELOPE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	EraseBinary (see page 909)	ApduInstruction for ERASE BINARY command as specified in ISO/IEC 7816-4:2005.
	EraseBinaryBerTlv (see page 909)	ApduInstruction for ERASE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	EraseRecord (see page 909)	ApduInstruction for ERASE RECORD (S) command as specified in ISO/IEC 7816-4:2005.
	ExternalAuthenticate (see page 909)	ApduInstruction for EXTERNAL (/ MUTUAL) AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.
	GeneralAuthenticate (see page 909)	ApduInstruction for GENERAL AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.
	GeneralAuthenticateBerTlv (see page 910)	ApduInstruction for GENERAL AUTHENTICATE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	GenerateAsymmetricKeyPair (see page 910)	ApduInstruction for GENERATE ASYMMETRIC KEY PAIR command as specified in ISO/IEC 7816-4:2005.
	GetChallenge (see page 910)	ApduInstruction for GET CHALLENGE command as specified in ISO/IEC 7816-4:2005.

	GetData (see page 910)	ApduInstruction for GET DATA command as specified in ISO/IEC 7816-4:2005.
	GetDataBerTlv (see page 910)	ApduInstruction for GET DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	GetResponse (see page 910)	ApduInstruction for GET RESPONSE command as specified in ISO/IEC 7816-4:2005.
	InternalAuthenticate (see page 910)	ApduInstruction for INTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.
	ManageChannel (see page 910)	ApduInstruction for MANAGE CHANNEL command as specified in ISO/IEC 7816-4:2005.
	ManageSecurityEnvironment (see page 911)	ApduInstruction for MANAGE SECURITY ENVIRONMENT command as specified in ISO/IEC 7816-4:2005.
	PerformScqlOperation (see page 911)	ApduInstruction for PERFORM SCQL OPERATION command as specified in ISO/IEC 7816-4:2005.
	PerformSecurityOperation (see page 911)	ApduInstruction for PERFORM SECURITY OPERATION command as specified in ISO/IEC 7816-4:2005.
	PerformTransactionOperation (see page 911)	ApduInstruction for PERFORM TRANSACTION OPERATION command as specified in ISO/IEC 7816-4:2005.
	PerformUserOperation (see page 911)	ApduInstruction for PERFORM USER OPERATION command as specified in ISO/IEC 7816-4:2005.
	PutData (see page 911)	ApduInstruction for PUT DATA command as specified in ISO/IEC 7816-4:2005.
	PutDataBerTlv (see page 911)	ApduInstruction for PUT DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	ReadBinary (see page 911)	ApduInstruction for READ BINARY command as specified in ISO/IEC 7816-4:2005.
	ReadBinaryBerTlv (see page 911)	ApduInstruction for READ BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	ReadRecord (see page 912)	ApduInstruction for READ RECORD (S) command as specified in ISO/IEC 7816-4:2005.
	ReadRecordBerTlv (see page 912)	ApduInstruction for READ RECORD (S) command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	ResetRetryCounter (see page 912)	ApduInstruction for RESET RETRY COUNTER command as specified in ISO/IEC 7816-4:2005.
	SearchBinary (see page 912)	ApduInstruction for SEARCH BINARY command as specified in ISO/IEC 7816-4:2005.
	SearchBinaryBerTlv (see page 912)	ApduInstruction for SEARCH BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	SearchRecord (see page 912)	ApduInstruction for SEARCH RECORD command as specified in ISO/IEC 7816-4:2005.
	Select (see page 912)	ApduInstruction for SELECT command as specified in ISO/IEC 7816-4:2005.
	TerminateCardUsage (see page 912)	ApduInstruction for TERMINATE CARD USAGE command as specified in ISO/IEC 7816-4:2005.
	TerminateDF (see page 913)	ApduInstruction for TERMINATE DF command as specified in ISO/IEC 7816-4:2005.
	TerminateEF (see page 913)	ApduInstruction for TERMINATE EF command as specified in ISO/IEC 7816-4:2005.
	UpdateBinary (see page 913)	ApduInstruction for UPDATE BINARY command as specified in ISO/IEC 7816-4:2005.
	UpdateBinaryBerTlv (see page 913)	ApduInstruction for UPDATE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	UpdateRecord (see page 913)	ApduInstruction for UPDATE RECORD command as specified in ISO/IEC 7816-4:2005.

	UpdateRecordBerTlv (see page 913)	ApduInstruction for UPDATE RECORD command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	Verify (see page 913)	ApduInstruction for VERIFY command as specified in ISO/IEC 7816-4:2005.
	VerifyBerTlv (see page 913)	ApduInstruction for VERIFY command with BER-TLV encode.
	WriteBinary (see page 913)	ApduInstruction for WRITE BINARY command as specified in ISO/IEC 7816-4:2005.
	WriteBinaryBerTlv (see page 914)	ApduInstruction for WRITE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.
	WriteRecord (see page 914)	ApduInstruction for WRITE RECORD command as specified in ISO/IEC 7816-4:2005.

Methods

	Name	Description
	ApduInstruction (see page 907)	Initializes a new instance of the ApduInstruction .

ApduInstruction Methods

	Name	Description
	Equals (see page 914)	Overloaded. Determines whether two Object instances are equal.
	GetHashCode (see page 914)	Serves as a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.
	ToString (see page 914)	Returns a String that represents the current Object.

ApduInstruction Operators

	Name	Description
	!= (see page 915)	Gets a value indicating whether the ApduInstruction is not identical to a second ApduInstruction.
	== (see page 915)	Gets a value indicating whether the ApduInstruction is identical to a second ApduInstruction.
	byte (see page 915)	Converts APDU instruction to byte.

ApduInstruction Properties

	Name	Description
	IsDataBerTlvEncoded (see page 915)	Indicates whether the data field of the APDU command with appropriate instruction byte (see page 915) (INS) is BER-TLV encoded.
	IsValid (see page 916)	Indicates whether APDU command instruction byte (see page 915) (INS) is valid, i.e. does not violate any of the INS encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.

6.2.8.1.2.1 ApduInstruction Operator

6.2.8.1.2.1.1 ApduInstruction.ApduInstruction Constructor (byte)

Initializes a new instance of the ApduInstruction .

C#

```
public ApduInstruction(byte value);
```

Parameters

Parameters	Description
byte value	byte (see page 915) value of the APDU instruction.

6.2.8.1.2.1.2 ApduInstruction.ApduInstruction Operator (byte)

Converts byte ([see page 915](#)) to APDU instruction.

C#

```
public static explicit operator AduInstruction(byte value);
```

Parameters

Parameters	Description
byte value	Byte value of the APDU instruction.

Returns

AduInstruction object.

6.2.8.1.2.2 AduInstruction Fields

6.2.8.1.2.2.1 AduInstruction.ActivateFile Field

AduInstruction (see page 904) for ACTIVATE FILE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ActivateFile = (AduInstruction)0x44;
```

6.2.8.1.2.2.2 AduInstruction.AppendRecord Field

AduInstruction (see page 904) for APPEND RECORD command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction AppendRecord = (AduInstruction)0xE2;
```

6.2.8.1.2.2.3 AduInstruction.ChangeReferenceData Field

AduInstruction (see page 904) for CHANGE REFERENCE DATA command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ChangeReferenceData = (AduInstruction)0x24;
```

6.2.8.1.2.2.4 AduInstruction.CreateFile Field

AduInstruction (see page 904) for CREATE FILE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction CreateFile = (AduInstruction)0xE0;
```

6.2.8.1.2.2.5 AduInstruction.DeactivateFile Field

AduInstruction (see page 904) for DEACTIVATE FILE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction DeactivateFile = (AduInstruction)0x04;
```

6.2.8.1.2.2.6 AduInstruction.DeleteFile Field

AduInstruction (see page 904) for DELETE FILE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction DeleteFile = (AduInstruction)0xE4;
```

6.2.8.1.2.2.7 AduInstruction.DisableVerificationRequirement Field

AduInstruction (see page 904) for DISABLE VERIFICATION REQUIREMENT command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction DisableVerificationRequirement =  
(AduInstruction)0x26;
```

6.2.8.1.2.2.8 AduInstruction.EnableVerificationRequirement Field

AduInstruction (see page 904) for ENABLE VERIFICATION REQUIREMENT command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction EnableVerificationRequirement =  
(AduInstruction)0x28;
```

6.2.8.1.2.2.9 AduInstruction.Envelope Field

AduInstruction (see page 904) for ENVELOPE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction Envelope = (AduInstruction)0xC2;
```

6.2.8.1.2.2.10 AduInstruction.EnvelopeBerTlv Field

AduInstruction (see page 904) for ENVELOPE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction EnvelopeBerTlv = (AduInstruction)0xC3;
```

6.2.8.1.2.2.11 AduInstruction.EraseBinary Field

AduInstruction (see page 904) for ERASE BINARY command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction EraseBinary = (AduInstruction)0x0E;
```

6.2.8.1.2.2.12 AduInstruction.EraseBinaryBerTlv Field

AduInstruction (see page 904) for ERASE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction EraseBinaryBerTlv = (AduInstruction)0x0F;
```

6.2.8.1.2.2.13 AduInstruction.EraseRecord Field

AduInstruction (see page 904) for ERASE RECORD (S) command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction EraseRecord = (AduInstruction)0x0C;
```

6.2.8.1.2.2.14 AduInstruction.ExternalAuthenticate Field

AduInstruction (see page 904) for EXTERNAL (/ MUTUAL) AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ExternalAuthenticate = (AduInstruction)0x82;
```

6.2.8.1.2.2.15 AduInstruction.GeneralAuthenticate Field

AduInstruction (see page 904) for GENERAL AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction GeneralAuthenticate = (ApmInstruction)0x86;
```

6.2.8.1.2.2.16 ApmInstruction.GeneralAuthenticateBerTlv Field

ApmInstruction (see page 904) for GENERAL AUTHENTICATE command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction GeneralAuthenticateBerTlv = (ApmInstruction)0x87;
```

6.2.8.1.2.2.17 ApmInstruction.GenerateAsymmetricKeyPair Field

ApmInstruction (see page 904) for GENERATE ASYMMETRIC KEY PAIR command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction GenerateAsymmetricKeyPair = (ApmInstruction)0x46;
```

6.2.8.1.2.2.18 ApmInstruction.GetChallenge Field

ApmInstruction (see page 904) for GET CHALLENGE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction GetChallenge = (ApmInstruction)0x84;
```

6.2.8.1.2.2.19 ApmInstruction.GetData Field

ApmInstruction (see page 904) for GET DATA command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction GetData = (ApmInstruction)0xCA;
```

6.2.8.1.2.2.20 ApmInstruction.GetDataBerTlv Field

ApmInstruction (see page 904) for GET DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction GetDataBerTlv = (ApmInstruction)0xCB;
```

6.2.8.1.2.2.21 ApmInstruction.GetResponse Field

ApmInstruction (see page 904) for GET RESPONSE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction GetResponse = (ApmInstruction)0xC0;
```

6.2.8.1.2.2.22 ApmInstruction.InternalAuthenticate Field

ApmInstruction (see page 904) for INTERNAL AUTHENTICATE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction InternalAuthenticate = (ApmInstruction)0x88;
```

6.2.8.1.2.2.23 ApmInstruction.ManageChannel Field

ApmInstruction (see page 904) for MANAGE CHANNEL command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApmInstruction ManageChannel = (ApmInstruction)0x70;
```

6.2.8.1.2.2.24 AduInstruction.ManageSecurityEnvironment Field

AduInstruction (see page 904) for MANAGE SECURITY ENVIRONMENT command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ManageSecurityEnvironment = (AduInstruction)0x22;
```

6.2.8.1.2.2.25 AduInstruction.PerformScqlOperation Field

AduInstruction (see page 904) for PERFORM SCQL OPERATION command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction PerformScqlOperation = (AduInstruction)0x10;
```

6.2.8.1.2.2.26 AduInstruction.PerformSecurityOperation Field

AduInstruction (see page 904) for PERFORM SECURITY OPERATION command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction PerformSecurityOperation = (AduInstruction)0x2A;
```

6.2.8.1.2.2.27 AduInstruction.PerformTransactionOperation Field

AduInstruction (see page 904) for PERFORM TRANSACTION OPERATION command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction PerformTransactionOperation = (AduInstruction)0x12;
```

6.2.8.1.2.2.28 AduInstruction.PerformUserOperation Field

AduInstruction (see page 904) for PERFORM USER OPERATION command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction PerformUserOperation = (AduInstruction)0x14;
```

6.2.8.1.2.2.29 AduInstruction.PutData Field

AduInstruction (see page 904) for PUT DATA command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction PutData = (AduInstruction)0xDA;
```

6.2.8.1.2.2.30 AduInstruction.PutDataBerTlv Field

AduInstruction (see page 904) for PUT DATA command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction PutDataBerTlv = (AduInstruction)0xDB;
```

6.2.8.1.2.2.31 AduInstruction.ReadBinary Field

AduInstruction (see page 904) for READ BINARY command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ReadBinary = (AduInstruction)0xB0;
```

6.2.8.1.2.2.32 AduInstruction.ReadBinaryBerTlv Field

AduInstruction (see page 904) for READ BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ReadBinaryBerTlv = (AduInstruction)0xB1;
```

6.2.8.1.2.2.33 AduInstruction.ReadRecord Field

AduInstruction (see page 904) for READ RECORD (S) command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ReadRecord = (AduInstruction)0xB2;
```

6.2.8.1.2.2.34 AduInstruction.ReadRecordBerTlv Field

AduInstruction (see page 904) for READ RECORD (S) command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ReadRecordBerTlv = (AduInstruction)0xB3;
```

6.2.8.1.2.2.35 AduInstruction.ResetRetryCounter Field

AduInstruction (see page 904) for RESET RETRY COUNTER command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction ResetRetryCounter = (AduInstruction)0x2C;
```

6.2.8.1.2.2.36 AduInstruction.SearchBinary Field

AduInstruction (see page 904) for SEARCH BINARY command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction SearchBinary = (AduInstruction)0xA0;
```

6.2.8.1.2.2.37 AduInstruction.SearchBinaryBerTlv Field

AduInstruction (see page 904) for SEARCH BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction SearchBinaryBerTlv = (AduInstruction)0xA1;
```

6.2.8.1.2.2.38 AduInstruction.SearchRecord Field

AduInstruction (see page 904) for SEARCH RECORD command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction SearchRecord = (AduInstruction)0xA2;
```

6.2.8.1.2.2.39 AduInstruction.Select Field

AduInstruction (see page 904) for SELECT command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction Select = (AduInstruction)0xA4;
```

6.2.8.1.2.2.40 AduInstruction.TerminateCardUsage Field

AduInstruction (see page 904) for TERMINATE CARD USAGE command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction TerminateCardUsage = (AduInstruction)0xFE;
```

6.2.8.1.2.2.41 AduInstruction.TerminateDF Field

AduInstruction (see page 904) for TERMINATE DF command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction TerminateDF = (AduInstruction)0xE6;
```

6.2.8.1.2.2.42 AduInstruction.TerminateEF Field

AduInstruction (see page 904) for TERMINATE EF command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction TerminateEF = (AduInstruction)0xE8;
```

6.2.8.1.2.2.43 AduInstruction.UpdateBinary Field

AduInstruction (see page 904) for UPDATE BINARY command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction UpdateBinary = (AduInstruction)0xD6;
```

6.2.8.1.2.2.44 AduInstruction.UpdateBinaryBerTlv Field

AduInstruction (see page 904) for UPDATE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction UpdateBinaryBerTlv = (AduInstruction)0xD7;
```

6.2.8.1.2.2.45 AduInstruction.UpdateRecord Field

AduInstruction (see page 904) for UPDATE RECORD command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction UpdateRecord = (AduInstruction)0xDC;
```

6.2.8.1.2.2.46 AduInstruction.UpdateRecordBerTlv Field

AduInstruction (see page 904) for UPDATE RECORD command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction UpdateRecordBerTlv = (AduInstruction)0xDD;
```

6.2.8.1.2.2.47 AduInstruction.Verify Field

AduInstruction (see page 904) for VERIFY command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction Verify = (AduInstruction)0x20;
```

6.2.8.1.2.2.48 AduInstruction.VerifyBerTlv Field

AduInstruction (see page 904) for VERIFY command with BER-TLV encode.

C#

```
public static readonly AduInstruction VerifyBerTlv = (AduInstruction)0x21;
```

6.2.8.1.2.2.49 AduInstruction.WriteBinary Field

AduInstruction (see page 904) for WRITE BINARY command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction WriteBinary = (AduInstruction)0xD0;
```

6.2.8.1.2.2.50 AduInstruction.WriteBinaryBerTlv Field

AduInstruction (see page 904) for WRITE BINARY command with BER-TLV encoded data as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction WriteBinaryBerTlv = (AduInstruction)0xD1;
```

6.2.8.1.2.2.51 AduInstruction.WriteRecord Field

AduInstruction (see page 904) for WRITE RECORD command as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduInstruction WriteRecord = (AduInstruction)0xD2;
```

6.2.8.1.2.3 AduInstruction Methods

6.2.8.1.2.3.1 AduInstruction.Equals Method

Overloaded. Determines whether two Object instances are equal.

C#

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

Parameters

Parameters	Description
object obj	The object to compare with the current object.

Returns

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

6.2.8.1.2.3.2 AduInstruction.GetHashCode Method

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

C#

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

Returns

A hash code for the current Object.

6.2.8.1.2.3.3 AduInstruction.ToString Method

Returns a String that represents the current Object.

C#

```
public override string ToString();
```

Returns

Returns a String that represents the current Object.

6.2.8.1.2.4 AduInstruction Operators

6.2.8.1.2.4.1 AduInstruction.!= Operator

Gets a value indicating whether the AduInstruction (see page 904) is not identical to a second AduInstruction (see page 904).

C#

```
public static bool operator !=(AduInstruction v1, AduInstruction v2);
```

Parameters

Parameters	Description
AduInstruction v1	AduInstruction (see page 904) object.
AduInstruction v2	AduInstruction (see page 904) object.

Returns

true if its two operands refer to different objects; otherwise, false.

6.2.8.1.2.4.2 AduInstruction.== Operator

Gets a value indicating whether the AduInstruction (see page 904) is identical to a second AduInstruction (see page 904).

C#

```
public static bool operator ==(AduInstruction v1, AduInstruction v2);
```

Parameters

Parameters	Description
AduInstruction v1	AduInstruction (see page 904) object.
AduInstruction v2	AduInstruction (see page 904) object.

Returns

true if the AduInstruction (see page 904) is identical to a second AduInstruction (see page 904); otherwise, false.

6.2.8.1.2.4.3 AduInstruction.byte Operator

Converts APDU instruction to byte.

C#

```
public static explicit operator byte(AduInstruction value);
```

Parameters

Parameters	Description
AduInstruction value	AduInstruction (see page 904) object.

Returns

AduInstruction (see page 904) converted to byte.

6.2.8.1.2.5 AduInstruction Properties

6.2.8.1.2.5.1 AduInstruction.IsDataBerTlvEncoded Property

Indicates whether the data field of the APDU command with appropriate instruction byte (see page 915) (INS) is BER-TLV encoded.

C#

```
public bool IsDataBerTlvEncoded;
```

Property value

true if the data field of the APDU command with appropriate instruction byte (see page 915) (INS) is BER-TLV encoded; otherwise, false.

6.2.8.1.2.5.2 AduInstruction.IsValid Property

Indicates whether APDU command instruction byte (see page 915) (INS) is valid, i.e. does not violate any of the INS encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.

C#

```
public bool IsValid;
```

Property value

true if APDU command instruction byte (see page 915) (INS) is valid; otherwise, false.

6.2.8.1.3 AduStatus Structure

Represents APDU response status.

C#

```
[Serializable]
public struct AduStatus {
    public static readonly AduStatus NoQualification = (AduStatus)0x9000;
    public static readonly ushort Proprietary9XXX = 0x9000;
    public static readonly ushort DataStillAvailableXX = 0x6100;
    public static readonly ushort WarningSonvmuXX = 0x6200;
    public static readonly AduStatus WarningSonvmuNoInformation = (AduStatus)0x6200;
    public static readonly ushort WarningSonvmuTriggeringByTheCardStart = 0x6202;
    public static readonly ushort WarningSonvmuTriggeringByTheCardEnd = 0x6280;
    public static readonly AduStatus WarningSonvmuPartOfReturnedDataMayBeCorrupted =
(AduStatus)0x6281;
    public static readonly AduStatus WarningSonvmuEndOfFileOrRecordBeforeReadingNBytes =
(AduStatus)0x6282;
    public static readonly AduStatus WarningSonvmuSelectedFileDeactivated =
(AduStatus)0x6283;
    public static readonly AduStatus WarningSonvmuFileControlInformationIncorrectlyFormatted
= (AduStatus)0x6284;
    public static readonly AduStatus WarningSonvmuSelectedFileInTerminationState =
(AduStatus)0x6285;
    public static readonly AduStatus WarningSonvmuNoInputDataAvailableFromASensorOnTheCard =
(AduStatus)0x6286;
    public static readonly ushort WarningSonvmcXX = 0x6300;
    public static readonly AduStatus WarningSonvmcNoInformation = (AduStatus)0x6300;
    public static readonly AduStatus WarningSonvmcFileFilledUpByTheLastWrite =
(AduStatus)0x6381;
    public static readonly ushort WarningSonvmcCounterX = 0x63C0;
    public static readonly ushort ErrorSonvmuXX = 0x6400;
    public static readonly AduStatus ErrorSonvmuExecutionError = (AduStatus)0x6400;
    public static readonly AduStatus ErrorSonvmuImmediateResponseRequiredByTheCard =
(AduStatus)0x6401;
    public static readonly ushort ErrorSonvmuTriggeringByTheCardStart = 0x6402;
    public static readonly ushort ErrorSonvmuTriggeringByTheCardEnd = 0x6480;
    public static readonly ushort ErrorSonvmcXX = 0x6500;
    public static readonly AduStatus ErrorSonvmcNoInformation = (AduStatus)0x6500;
    public static readonly AduStatus ErrorSonvmcMemoryFailure = (AduStatus)0x6581;
    public static readonly ushort ErrorSecurityXX = 0x6600;
    public static readonly AduStatus ErrorWrongLength = (AduStatus)0x6700;
    public static readonly ushort ErrorProprietary67XX = 0x6700;
    public static readonly ushort ErrorFicnsXX = 0x6800;
    public static readonly AduStatus ErrorFicnsNoInformation = (AduStatus)0x6800;
    public static readonly AduStatus ErrorFicnsLogicalChannelNotSupported =
(AduStatus)0x6881;
    public static readonly AduStatus ErrorFicnsSecureMessagingNotSupported =
```

```

(ApduStatus)0x6882;
    public static readonly ApduStatus ErrorFicnsLastCommandOfTheChainExpected =
(ApduStatus)0x6883;
    public static readonly ApduStatus ErrorFicnsCommandChainingNotSupported =
(ApduStatus)0x6884;
    public static readonly ushort ErrorCnaXX = 0x6900;
    public static readonly ApduStatus ErrorCnaNoInformation = (ApduStatus)0x6900;
    public static readonly ApduStatus ErrorCnaCommandIncompatibleWithFileStructure =
(ApduStatus)0x6981;
    public static readonly ApduStatus ErrorCnaSecurityStatusNotSatisfied = (ApduStatus)0x6982;
    public static readonly ApduStatus ErrorCnaAuthenticationMethodBlocked =
(ApduStatus)0x6983;
    public static readonly ApduStatus ErrorCnaReferenceDataNotUsable = (ApduStatus)0x6984;
    public static readonly ApduStatus ErrorCnaConditionsOfUseNotSatisfied =
(ApduStatus)0x6985;
    public static readonly ApduStatus ErrorCnaCommandNotAllowed = (ApduStatus)0x6986;
    public static readonly ApduStatus ErrorCnaExpectedSecureMessagingDataObjectsMissing =
(ApduStatus)0x6987;
    public static readonly ApduStatus ErrorCnaIncorrectSecureMessagingDataObjects =
(ApduStatus)0x6988;
    public static readonly ushort ErrorWP1P2XX = 0x6A00;
    public static readonly ApduStatus ErrorWP1P2NoInformation = (ApduStatus)0x6A00;
    public static readonly ApduStatus ErrorWP1P2IncorrectParametersInTheCommandDataField =
(ApduStatus)0x6A80;
    public static readonly ApduStatus ErrorWP1P2FunctionNotSupported = (ApduStatus)0x6A81;
    public static readonly ApduStatus ErrorWP1P2FileOrApplicationNotFound =
(ApduStatus)0x6A82;
    public static readonly ApduStatus ErrorWP1P2RecordNotFound = (ApduStatus)0x6A83;
    public static readonly ApduStatus ErrorWP1P2NotEnoughMemorySpaceInTheFile =
(ApduStatus)0x6A84;
    public static readonly ApduStatus ErrorWP1P2NcInconsistentWithTlvStructure =
(ApduStatus)0x6A85;
    public static readonly ApduStatus ErrorWP1P2IncorrectParametersP1P2 = (ApduStatus)0x6A86;
    public static readonly ApduStatus ErrorWP1P2NcInconsistentWithParametersP1P2 =
(ApduStatus)0x6A87;
    public static readonly ApduStatus ErrorWP1P2ReferencedDataOrReferenceDataNotFound =
(ApduStatus)0x6A88;
    public static readonly ApduStatus ErrorWP1P2FileAlreadyExists = (ApduStatus)0x6A89;
    public static readonly ApduStatus ErrorWP1P2DFNameAlreadyExists = (ApduStatus)0x6A8A;
    public static readonly ApduStatus ErrorWP1P2NoQualification = (ApduStatus)0x6B00;
    public static readonly ushort ErrorProprietary6BXX = 0x6B00;
    public static readonly ushort ErrorWrongLexX = 0x6C00;
    public static readonly ApduStatus ErrorInstructionNotSupportedOrInvalid =
(ApduStatus)0x6D00;
    public static readonly ushort ErrorProprietary6DXX = 0x6D00;
    public static readonly ApduStatus ErrorClassNotSupported = (ApduStatus)0x6E00;
    public static readonly ushort ErrorProprietary6EXX = 0x6E00;
    public static readonly ApduStatus ErrorNoPreciseDiagnosis = (ApduStatus)0x6F00;
    public static readonly ushort ErrorProprietary6FXX = 0x6F00;
}

```

ApduStatus Fields

	Name	Description
	DataStillAvailableXX (see page 925)	Mask for ApduStatus meaning normal processing (SW2 (see page 941) encodes the number of data bytes still available) as specified in ISO/IEC 7816-4:2005.
	ErrorClassNotSupported (see page 925)	ApduStatus meaning checking error class not supported as specified in ISO/IEC 7816-4:2005.
	ErrorCnaAuthenticationMethodBlocked (see page 925)	ApduStatus meaning checking error command not allowed, authentication method blocked as specified in ISO/IEC 7816-4:2005.

	ErrorCnaCommandIncompatibleWithFileStructure (see page 925)	ApduStatus meaning checking error command not allowed, command incompatible with file structure as specified in ISO/IEC 7816-4:2005.
	ErrorCnaCommandNotAllowed (see page 925)	ApduStatus meaning checking error command not allowed, command not allowed (no current EF) as specified in ISO/IEC 7816-4:2005.
	ErrorCnaConditionsOfUseNotSatisfied (see page 925)	ApduStatus meaning checking error command not allowed, conditions of use not satisfied as specified in ISO/IEC 7816-4:2005.
	ErrorCnaExpectedSecureMessagingDataObjectsMissing (see page 925)	ApduStatus meaning checking error command not allowed, expected secure messaging data objects missing as specified in ISO/IEC 7816-4:2005.
	ErrorCnaIncorrectSecureMessagingDataObjects (see page 925)	ApduStatus meaning checking error command not allowed, incorrect secure messaging data objects as specified in ISO/IEC 7816-4:2005.
	ErrorCnaNoInformation (see page 926)	ApduStatus meaning checking error command not allowed, no information given as specified in ISO/IEC 7816-4:2005.
	ErrorCnaReferenceDataNotUsable (see page 926)	ApduStatus meaning checking error command not allowed, reference data not usable as specified in ISO/IEC 7816-4:2005.
	ErrorCnaSecurityStatusNotSatisfied (see page 926)	ApduStatus meaning checking error command not allowed, security status not satisfied as specified in ISO/IEC 7816-4:2005.
	ErrorCnaXX (see page 926)	Mask for ApduStatus meaning checking error command not allowed (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.
	ErrorFicnsCommandChainingNotSupported (see page 926)	ApduStatus meaning checking error functions in CLA not supported, command chaining not supported as specified in ISO/IEC 7816-4:2005.
	ErrorFicnsLastCommandOfTheChainExpected (see page 926)	ApduStatus meaning checking error functions in CLA not supported, last command of the chain expected as specified in ISO/IEC 7816-4:2005.
	ErrorFicnsLogicalChannelNotSupported (see page 926)	ApduStatus meaning checking error functions in CLA not supported, logical channel not supported as specified in ISO/IEC 7816-4:2005.
	ErrorFicnsNoInformation (see page 927)	ApduStatus meaning checking error functions in CLA not supported, no information given as specified in ISO/IEC 7816-4:2005.
	ErrorFicnsSecureMessagingNotSupported (see page 927)	ApduStatus meaning checking error functions in CLA not supported, secure messaging not supported as specified in ISO/IEC 7816-4:2005.
	ErrorFicnsXX (see page 927)	Mask for ApduStatus meaning checking error functions in CLA not supported (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

	ErrorInstructionNotSupportedOrInvalid (see page 927)	ApduStatus meaning checking error instruction code not supported or invalid as specified in ISO/IEC 7816-4:2005.
	ErrorNoPreciseDiagnosis (see page 927)	ApduStatus meaning checking error no precise diagnosis as specified in ISO/IEC 7816-4:2005.
	ErrorProprietary67XX (see page 927)	Mask for ApduStatus meaning proprietary checking error 67XX as specified in ISO/IEC 7816-4:2005.
	ErrorProprietary6BXX (see page 927)	Mask for ApduStatus meaning proprietary checking error 6BXX as specified in ISO/IEC 7816-4:2005.
	ErrorProprietary6DXX (see page 927)	Mask for ApduStatus meaning proprietary checking error 6DXX as specified in ISO/IEC 7816-4:2005.
	ErrorProprietary6EXX (see page 928)	Mask for ApduStatus meaning proprietary checking error 6EXX as specified in ISO/IEC 7816-4:2005.
	ErrorProprietary6FXX (see page 928)	Mask for ApduStatus meaning proprietary checking error 6FXX as specified in ISO/IEC 7816-4:2005.
	ErrorSecurityXX (see page 928)	Mask for ApduStatus meaning execution error security-related issues as specified in ISO/IEC 7816-4:2005.
	ErrorSonvmcMemoryFailure (see page 928)	ApduStatus meaning memory failure as specified in ISO/IEC 7816-4:2005.
	ErrorSonvmcNoInformation (see page 928)	ApduStatus meaning execution error state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.
	ErrorSonvmcXX (see page 928)	Mask for ApduStatus meaning execution error state of non-volatile memory has changed (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.
	ErrorSonvmuExecutionError (see page 928)	ApduStatus meaning execution error state of non-volatile memory is unchanged, execution error as specified in ISO/IEC 7816-4:2005.
	ErrorSonvmuImmediateResponseRequiredByTheCard (see page 928)	ApduStatus meaning execution error state of non-volatile memory is unchanged, immediate response required by the card as specified in ISO/IEC 7816-4:2005.
	ErrorSonvmuTriggeringByTheCardEnd (see page 929)	End of values for ApduStatus meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.
	ErrorSonvmuTriggeringByTheCardStart (see page 929)	Start of values for ApduStatus meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.
	ErrorSonvmuXX (see page 929)	Mask for ApduStatus meaning execution error state of non-volatile memory is unchanged (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

	ErrorWP1P2DFNameAlreadyExists (see page 929)	ApduStatus meaning checking error wrong parameters P1-P2, DF name already exists as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2FileAlreadyExists (see page 929)	ApduStatus meaning checking error wrong parameters P1-P2, file already exists as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2FileOrApplicationNotFound (see page 929)	ApduStatus meaning checking error wrong parameters P1-P2, file or application not found as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2FunctionNotSupported (see page 929)	ApduStatus meaning checking error wrong parameters P1-P2, function not supported as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2IncorrectParametersInTheCommandDataField (see page 929)	ApduStatus meaning checking error wrong parameters P1-P2, incorrect parameters in the command data field as specified in ISO/IEC 7816-4:2005.\
	ErrorWP1P2IncorrectParametersP1P2 (see page 930)	ApduStatus meaning checking error wrong parameters P1-P2, incorrect parameters P1-P2 as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2NcInconsistentWithParametersP1P2 (see page 930)	ApduStatus meaning checking error wrong parameters P1-P2, Nc inconsistent with parameters P1-P2 as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2NcInconsistentWithTlvStructure (see page 930)	ApduStatus meaning checking error wrong parameters P1-P2, Nc inconsistent with TLV structure as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2NoInformation (see page 930)	ApduStatus meaning checking error wrong parameters P1-P2, no information given as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2NoQualification (see page 930)	ApduStatus meaning checking error wrong parameters P1-P2, no further indication as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2NotEnoughMemorySpaceInTheFile (see page 930)	ApduStatus meaning checking error wrong parameters P1-P2, not enough memory space in the file as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2RecordNotFound (see page 930)	ApduStatus meaning checking error wrong parameters P1-P2, record not found as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2ReferencedDataOrReferenceDataNotFound (see page 931)	ApduStatus meaning checking error wrong parameters P1-P2, referenced data or reference data not found (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.
	ErrorWP1P2XX (see page 931)	Mask for ApduStatus meaning checking error wrong parameters P1-P2 (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.
	ErrorWrongLength (see page 931)	ApduStatus meaning checking error wrong length, no further indication as specified in ISO/IEC 7816-4:2005.
	ErrorWrongLeXX (see page 931)	Mask for ApduStatus meaning checking error wrong Le field (SW2 (see page 941) encodes the exact number of available data bytes) as specified in ISO/IEC 7816-4:2005.

	NoQualification (see page 931)	ApduStatus meaning normal processing, no further qualification as specified in ISO/IEC 7816-4:2005.
	Proprietary9XXX (see page 931)	Mask for ApduStatus meaning proprietary status 9XXX as specified in ISO/IEC 7816-4:2005.
	WarningSonvmcCounterX (see page 931)	Mask for ApduStatus meaning warning processing state of non-volatile memory has changed, counter from 0 to 15 encoded by 'X' (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.
	WarningSonvmcFileFilledUpByTheLastWrite (see page 931)	ApduStatus meaning warning processing state of non-volatile memory has changed, file filled up by the last write as specified in ISO/IEC 7816-4:2005.
	WarningSonvmcNoInformation (see page 932)	ApduStatus meaning warning processing state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.
	WarningSonvmcXX (see page 932)	Mask for ApduStatus meaning warning processing state of non-volatile memory has changed (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuEndOfFileOrRecordBeforeReadingNeBytes (see page 932)	ApduStatus meaning warning processing state of non-volatile memory is unchanged, end of file or record reached before reading Ne bytes as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuFileControllInformationIncorrectlyFormatted (see page 932)	ApduStatus meaning warning processing state of non-volatile memory is unchanged, file control information not formatted according to 5.3.3 as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuNoInformation (see page 932)	ApduStatus meaning warning processing state of non-volatile memory is unchanged, no information given as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuNoInputDataAvailableFromASensorOnTheCard (see page 932)	ApduStatus meaning warning processing state of non-volatile memory is unchanged, no input data available from a sensor on the card as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuPartOfReturnedDataMayBeCorrupted (see page 932)	ApduStatus meaning warning processing state of non-volatile memory is unchanged, part of returned data may be corrupted as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuSelectedFileDeactivated (see page 933)	ApduStatus meaning warning processing state of non-volatile memory is unchanged, selected file deactivated as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuSelectedFileInTerminationState (see page 933)	ApduStatus meaning warning processing state of non-volatile memory is unchanged, selected file in termination state as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuTriggeringByTheCardEnd (see page 933)	Start of values for ApduStatus meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

	WarningSonvmuTriggeringByTheCardStart (see page 933)	End of values for ApduStatus meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.
	WarningSonvmuXX (see page 933)	Mask for ApduStatus meaning warning processing state of non-volatile memory is unchanged (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

Methods

	Name	Description
	ApduStatus (see page 924)	Initializes a new instance of the ApduStatus .

ApduStatus Methods

	Name	Description
	Equals (see page 933)	Overloaded. Determines whether two Object instances are equal.
	GetHashCode (see page 933)	Serves as a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.
	ToString (see page 934)	Returns a String that represents the current Object.

ApduStatus Operators

	Name	Description
	!= (see page 934)	Gets a value indicating whether the ApduStatus is not identical to a second ApduStatus.
	== (see page 934)	Gets a value indicating whether the ApduStatus is identical to a second ApduStatus.
	ushort (see page 934)	Converts APDU status to ushort.

ApduStatus Properties

	Name	Description
	DataStillAvailableValue (see page 935)	Gets the number of available data bytes from the APDU response status.
	ErrorSonvmuTriggeringByTheCardValue (see page 935)	Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 64XX.
	ErrorWrongLeValue (see page 935)	Retrieves the exact number of available data bytes from the APDU response status. Applicable only if APDU status is 6CXX.
	IsCheckingError (see page 935)	Retrieves value indicating whether APDU response status means checking error.
	IsDataStillAvailable (see page 935)	Retrieves a value indicating whether APDU status means normal processing (SW2 (see page 941) encodes the number of data bytes still available).
	IsErrorCna (see page 936)	Retrieves a value indicating whether APDU status means checking error command not allowed (further qualification in SW2 (see page 941)).
	IsErrorFicns (see page 936)	Retrieves a value indicating whether APDU status means checking error functions in CLA not supported (further qualification in SW2 (see page 941)).
	IsErrorProprietary67XX (see page 936)	Retrieves a value indicating whether APDU status means proprietary error 67XX.
	IsErrorProprietary6BXX (see page 936)	Retrieves a value indicating whether APDU status means proprietary error 6BXX.

	IsErrorProprietary6DXX (see page 936)	Retrieves a value indicating whether APDU status means proprietary error 6DXX.
	IsErrorProprietary6EXX (see page 937)	Retrieves a value indicating whether APDU status means proprietary error 6EXX.
	IsErrorProprietary6FXX (see page 937)	Retrieves a value indicating whether APDU status means proprietary error 6FXX.
	IsErrorSecurity (see page 937)	Retrieves a value indicating whether APDU status means execution error security-related issues.
	IsErrorSonvmc (see page 937)	Retrieves a value indicating whether APDU status means execution error state of nonvolatile memory has changed (further qualification in SW2 (see page 941)).
	IsErrorSonvmu (see page 937)	Retrieves a value indicating whether APDU status means execution error state of nonvolatile memory is unchanged (further qualification in SW2 (see page 941)).
	IsErrorSonvmuTriggeringByTheCard (see page 937)	Retrieves a value indicating whether APDU status means execution error state of nonvolatile memory is unchanged, triggering by the card.
	IsErrorWP1P2 (see page 938)	Retrieves a value indicating whether APDU status means checking error wrong parameters P1-P2 (further qualification in SW2 (see page 941)).
	IsErrorWrongLe (see page 938)	Retrieves a value indicating whether APDU status means checking error wrong Le field (SW2 (see page 941) encodes the exact number of available data bytes).
	IsExecutionError (see page 938)	Retrieves value indicating whether APDU response status means execution error.
	IsInterindustry (see page 938)	Retrieves value indicating whether APDU response status is interindustry.
	IsNormalProcessing (see page 938)	Retrieves value indicating whether APDU response status means normal processing.
	IsProcessAborted (see page 939)	Retrieves value indicating whether APDU response status means process aborted.
	IsProcessComplete (see page 939)	Retrieves value indicating whether APDU response status means process completed.
	IsProprietary (see page 939)	Retrieves value indicating whether APDU response status is proprietary.
	IsProprietary9XXX (see page 939)	Retrieves a value indicating whether APDU status means proprietary status 9XXX.
	IsStateOfNonVolatileMemoryChanged (see page 939)	Retrieves value indicating whether APDU response status means state of non-volatile memory has changed.
	IsStateOfNonVolatileMemoryUnchanged (see page 939)	Retrieves value indicating whether APDU response status means state of non-volatile memory has changed state of non-volatile memory is unchanged.
	IsValid (see page 940)	Retrieves value indicating whether APDU response status is valid, i.e. does not violate any of the SW1SW2 status bytes encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.
	IsWarningProcessing (see page 940)	Retrieves value indicating whether APDU response status means warning processing.
	IsWarningSonvmc (see page 940)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed (further qualification in SW2 (see page 941)).
	IsWarningSonvmcCounter (see page 940)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed, counter from 0 to 15 (exact meaning depending on the command).

	IsWarningSonvmu (see page 940)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged (further qualification in SW2 (see page 941)).
	IsWarningSonvmuTriggeringByTheCard (see page 941)	Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged, triggering by the card.
	SW1 (see page 941)	Retrieves the first byte of the specified APDU status.
	SW2 (see page 941)	Retrieves the second byte of the specified APDU status.
	WarningSonvmcCounterValue (see page 941)	Retrieves the value of a counter returned by the card from the APDU response status. Applicable only if APDU status is 63CX.
	WarningSonvmuTriggeringByTheCardValue (see page 941)	Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 62XX.

6.2.8.1.3.1 ApmduStatus Operator

6.2.8.1.3.1.1 ApmduStatus.ApmduStatus Constructor (byte, byte)

Initializes a new instance of the ApmduStatus .

C#

```
public ApmduStatus(byte sw1, byte sw2);
```

Parameters

Parameters	Description
byte sw1	First byte of processing state.
byte sw2	Second byte of processing state.

6.2.8.1.3.1.2 ApmduStatus.ApmduStatus Constructor (ushort)

Initializes a new instance of the ApmduStatus .

C#

```
public ApmduStatus(ushort value);
```

Parameters

Parameters	Description
ushort value	Indicates the processing state.

6.2.8.1.3.1.3 ApmduStatus.ApmduStatus Operator (ushort)

Converts byte to APDU status.

C#

```
public static explicit operator ApmduStatus(ushort value);
```

Parameters

Parameters	Description
ushort value	ushort (see page 934) value of the APDU status.

Returns

ApmduStatus object.

6.2.8.1.3.2 ApmduStatus Fields

6.2.8.1.3.2.1 ApduStatus.DataStillAvailableXX Field

Mask for ApduStatus (see page 916) meaning normal processing (SW2 (see page 941) encodes the number of data bytes still available) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort DataStillAvailableXX = 0x6100;
```

6.2.8.1.3.2.2 ApduStatus.ErrorClassNotSupported Field

ApduStatus (see page 916) meaning checking error class not supported as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorClassNotSupported = (ApduStatus)0x6E00;
```

6.2.8.1.3.2.3 ApduStatus.ErrorCnaAuthenticationMethodBlocked Field

ApduStatus (see page 916) meaning checking error command not allowed, authentication method blocked as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaAuthenticationMethodBlocked = (ApduStatus)0x6983;
```

6.2.8.1.3.2.4 ApduStatus.ErrorCnaCommandIncompatibleWithFileStructure Field

ApduStatus (see page 916) meaning checking error command not allowed, command incompatible with file structure as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaCommandIncompatibleWithFileStructure =  
(ApduStatus)0x6981;
```

6.2.8.1.3.2.5 ApduStatus.ErrorCnaCommandNotAllowed Field

ApduStatus (see page 916) meaning checking error command not allowed, command not allowed (no current EF) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaCommandNotAllowed = (ApduStatus)0x6986;
```

6.2.8.1.3.2.6 ApduStatus.ErrorCnaConditionsOfUseNotSatisfied Field

ApduStatus (see page 916) meaning checking error command not allowed, conditions of use not satisfied as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaConditionsOfUseNotSatisfied = (ApduStatus)0x6985;
```

6.2.8.1.3.2.7 ApduStatus.ErrorCnaExpectedSecureMessagingDataObjectsMissing Field

ApduStatus (see page 916) meaning checking error command not allowed, expected secure messaging data objects missing as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaExpectedSecureMessagingDataObjectsMissing =  
(ApduStatus)0x6987;
```

6.2.8.1.3.2.8 ApduStatus.ErrorCnaIncorrectSecureMessagingDataObjects Field

ApduStatus (see page 916) meaning checking error command not allowed, incorrect secure messaging data objects as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaIncorrectSecureMessagingDataObjects =  
(ApduStatus)0x6988;
```

6.2.8.1.3.2.9 ApduStatus.ErrorCnaNoInformation Field

ApduStatus (see page 916) meaning checking error command not allowed, no information given as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaNoInformation = (ApduStatus)0x6900;
```

6.2.8.1.3.2.10 ApduStatus.ErrorCnaReferenceDataNotUsable Field

ApduStatus (see page 916) meaning checking error command not allowed, reference data not usable as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaReferenceDataNotUsable = (ApduStatus)0x6984;
```

6.2.8.1.3.2.11 ApduStatus.ErrorCnaSecurityStatusNotSatisfied Field

ApduStatus (see page 916) meaning checking error command not allowed, security status not satisfied as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorCnaSecurityStatusNotSatisfied = (ApduStatus)0x6982;
```

6.2.8.1.3.2.12 ApduStatus.ErrorCnaXX Field

Mask for ApduStatus (see page 916) meaning checking error command not allowed (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorCnaXX = 0x6900;
```

6.2.8.1.3.2.13 ApduStatus.ErrorFicnsCommandChainingNotSupported Field

ApduStatus (see page 916) meaning checking error functions in CLA not supported, command chaining not supported as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorFicnsCommandChainingNotSupported =  
(ApduStatus)0x6884;
```

6.2.8.1.3.2.14 ApduStatus.ErrorFicnsLastCommandOfTheChainExpected Field

ApduStatus (see page 916) meaning checking error functions in CLA not supported, last command of the chain expected as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorFicnsLastCommandOfTheChainExpected =  
(ApduStatus)0x6883;
```

6.2.8.1.3.2.15 ApduStatus.ErrorFicnsLogicalChannelNotSupported Field

ApduStatus (see page 916) meaning checking error functions in CLA not supported, logical channel not supported as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorFicnsLogicalChannelNotSupported = (ApduStatus)0x6881;
```

6.2.8.1.3.2.16 ApduStatus.ErrorFicnsNoInformation Field

ApduStatus (see page 916) meaning checking error functions in CLA not supported, no information given as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorFicnsNoInformation = (ApduStatus)0x6800;
```

6.2.8.1.3.2.17 ApduStatus.ErrorFicnsSecureMessagingNotSupported Field

ApduStatus (see page 916) meaning checking error functions in CLA not supported, secure messaging not supported as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorFicnsSecureMessagingNotSupported =  
(ApduStatus)0x6882;
```

6.2.8.1.3.2.18 ApduStatus.ErrorFicnsXX Field

Mask for ApduStatus (see page 916) meaning checking error functions in CLA not supported (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorFicnsXX = 0x6800;
```

6.2.8.1.3.2.19 ApduStatus.ErrorInstructionNotSupportedOrInvalid Field

ApduStatus (see page 916) meaning checking error instruction code not supported or invalid as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorInstructionNotSupportedOrInvalid =  
(ApduStatus)0x6D00;
```

6.2.8.1.3.2.20 ApduStatus.ErrorNoPreciseDiagnosis Field

ApduStatus (see page 916) meaning checking error no precise diagnosis as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorNoPreciseDiagnosis = (ApduStatus)0x6F00;
```

6.2.8.1.3.2.21 ApduStatus.ErrorProprietary67XX Field

Mask for ApduStatus (see page 916) meaning proprietary checking error 67XX as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorProprietary67XX = 0x6700;
```

6.2.8.1.3.2.22 ApduStatus.ErrorProprietary6BXX Field

Mask for ApduStatus (see page 916) meaning proprietary checking error 6BXX as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorProprietary6BXX = 0x6B00;
```

6.2.8.1.3.2.23 ApduStatus.ErrorProprietary6DXX Field

Mask for ApduStatus (see page 916) meaning proprietary checking error 6DXX as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorProprietary6DXX = 0x6D00;
```


6.2.8.1.3.2.24 ApduStatus.ErrorProprietary6EXX Field

Mask for ApduStatus (see page 916) meaning proprietary checking error 6EXX as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorProprietary6EXX = 0x6E00;
```

6.2.8.1.3.2.25 ApduStatus.ErrorProprietary6FXX Field

Mask for ApduStatus (see page 916) meaning proprietary checking error 6FXX as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorProprietary6FXX = 0x6F00;
```

6.2.8.1.3.2.26 ApduStatus.ErrorSecurityXX Field

Mask for ApduStatus (see page 916) meaning execution error security-related issues as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorSecurityXX = 0x6600;
```

6.2.8.1.3.2.27 ApduStatus.ErrorSonvmcMemoryFailure Field

ApduStatus (see page 916) meaning memory failure as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorSonvmcMemoryFailure = (ApduStatus)0x6581;
```

6.2.8.1.3.2.28 ApduStatus.ErrorSonvmcNoInformation Field

ApduStatus (see page 916) meaning execution error state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorSonvmcNoInformation = (ApduStatus)0x6500;
```

6.2.8.1.3.2.29 ApduStatus.ErrorSonvmcXX Field

Mask for ApduStatus (see page 916) meaning execution error state of non-volatile memory has changed (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorSonvmcXX = 0x6500;
```

6.2.8.1.3.2.30 ApduStatus.ErrorSonvmuExecutionError Field

ApduStatus (see page 916) meaning execution error state of non-volatile memory is unchanged, execution error as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorSonvmuExecutionError = (ApduStatus)0x6400;
```

6.2.8.1.3.2.31 ApduStatus.ErrorSonvmuImmediateResponseRequiredByTheCard Field

ApduStatus (see page 916) meaning execution error state of non-volatile memory is unchanged, immediate response required by the card as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorSonvmuImmediateResponseRequiredByTheCard =  
(ApduStatus)0x6401;
```

6.2.8.1.3.2.32 ApduStatus.ErrorSonvmuTriggeringByTheCardEnd Field

End of values for ApduStatus (see page 916) meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorSonvmuTriggeringByTheCardEnd = 0x6480;
```

6.2.8.1.3.2.33 ApduStatus.ErrorSonvmuTriggeringByTheCardStart Field

Start of values for ApduStatus (see page 916) meaning execution error state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorSonvmuTriggeringByTheCardStart = 0x6402;
```

6.2.8.1.3.2.34 ApduStatus.ErrorSonvmuXX Field

Mask for ApduStatus (see page 916) meaning execution error state of non-volatile memory is unchanged (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorSonvmuXX = 0x6400;
```

6.2.8.1.3.2.35 ApduStatus.ErrorWP1P2DFNameAlreadyExists Field

ApduStatus (see page 916) meaning checking error wrong parameters P1-P2, DF name already exists as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorWP1P2DFNameAlreadyExists = (ApduStatus)0x6A8A;
```

6.2.8.1.3.2.36 ApduStatus.ErrorWP1P2FileAlreadyExists Field

ApduStatus (see page 916) meaning checking error wrong parameters P1-P2, file already exists as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorWP1P2FileAlreadyExists = (ApduStatus)0x6A89;
```

6.2.8.1.3.2.37 ApduStatus.ErrorWP1P2FileOrApplicationNotFound Field

ApduStatus (see page 916) meaning checking error wrong parameters P1-P2, file or application not found as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorWP1P2FileOrApplicationNotFound = (ApduStatus)0x6A82;
```

6.2.8.1.3.2.38 ApduStatus.ErrorWP1P2FunctionNotSupported Field

ApduStatus (see page 916) meaning checking error wrong parameters P1-P2, function not supported as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorWP1P2FunctionNotSupported = (ApduStatus)0x6A81;
```

6.2.8.1.3.2.39 ApduStatus.ErrorWP1P2IncorrectParametersInTheCommandDataField Field

ApduStatus (see page 916) meaning checking error wrong parameters P1-P2, incorrect parameters in the command data field as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduStatus ErrorWP1P2IncorrectParametersInTheCommandDataField =  
(AduStatus)0x6A80;
```

6.2.8.1.3.2.40 AduStatus.ErrorWP1P2IncorrectParametersP1P2 Field

AduStatus (see page 916) meaning checking error wrong parameters P1-P2, incorrect parameters P1-P2 as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduStatus ErrorWP1P2IncorrectParametersP1P2 = (AduStatus)0x6A86;
```

6.2.8.1.3.2.41 AduStatus.ErrorWP1P2NcInconsistentWithParametersP1P2 Field

AduStatus (see page 916) meaning checking error wrong parameters P1-P2, Nc inconsistent with parameters P1-P2 as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduStatus ErrorWP1P2NcInconsistentWithParametersP1P2 =  
(AduStatus)0x6A87;
```

6.2.8.1.3.2.42 AduStatus.ErrorWP1P2NcInconsistentWithTlvStructure Field

AduStatus (see page 916) meaning checking error wrong parameters P1-P2, Nc inconsistent with TLV structure as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduStatus ErrorWP1P2NcInconsistentWithTlvStructure =  
(AduStatus)0x6A85;
```

6.2.8.1.3.2.43 AduStatus.ErrorWP1P2NoInformation Field

AduStatus (see page 916) meaning checking error wrong parameters P1-P2, no information given as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduStatus ErrorWP1P2NoInformation = (AduStatus)0x6A00;
```

6.2.8.1.3.2.44 AduStatus.ErrorWP1P2NoQualification Field

AduStatus (see page 916) meaning checking error wrong parameters P1-P2, no further indication as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduStatus ErrorWP1P2NoQualification = (AduStatus)0x6B00;
```

6.2.8.1.3.2.45 AduStatus.ErrorWP1P2NotEnoughMemorySpaceInTheFile Field

AduStatus (see page 916) meaning checking error wrong parameters P1-P2, not enough memory space in the file as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduStatus ErrorWP1P2NotEnoughMemorySpaceInTheFile =  
(AduStatus)0x6A84;
```

6.2.8.1.3.2.46 AduStatus.ErrorWP1P2RecordNotFound Field

AduStatus (see page 916) meaning checking error wrong parameters P1-P2, record not found as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly AduStatus ErrorWP1P2RecordNotFound = (AduStatus)0x6A83;
```

6.2.8.1.3.2.47 ApduStatus.ErrorWP1P2ReferencedDataOrReferenceDataNotFound Field

ApduStatus (see page 916) meaning checking error wrong parameters P1-P2, referenced data or reference data not found (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorWP1P2ReferencedDataOrReferenceDataNotFound =  
(ApduStatus)0x6A88;
```

6.2.8.1.3.2.48 ApduStatus.ErrorWP1P2XX Field

Mask for ApduStatus (see page 916) meaning checking error wrong parameters P1-P2 (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorWP1P2XX = 0x6A00;
```

6.2.8.1.3.2.49 ApduStatus.ErrorWrongLength Field

ApduStatus (see page 916) meaning checking error wrong length, no further indication as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus ErrorWrongLength = (ApduStatus)0x6700;
```

6.2.8.1.3.2.50 ApduStatus.ErrorWrongLeXX Field

Mask for ApduStatus (see page 916) meaning checking error wrong Le field (SW2 (see page 941) encodes the exact number of available data bytes) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort ErrorWrongLeXX = 0x6C00;
```

6.2.8.1.3.2.51 ApduStatus.NoQualification Field

ApduStatus (see page 916) meaning normal processing, no further qualification as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus NoQualification = (ApduStatus)0x9000;
```

6.2.8.1.3.2.52 ApduStatus.Proprietary9XXX Field

Mask for ApduStatus (see page 916) meaning proprietary status 9XXX as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort Proprietary9XXX = 0x9000;
```

6.2.8.1.3.2.53 ApduStatus.WarningSonvmcCounterX Field

Mask for ApduStatus (see page 916) meaning warning processing state of non-volatile memory has changed, counter from 0 to 15 encoded by 'X' (exact meaning depending on the command) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort WarningSonvmcCounterX = 0x63C0;
```

6.2.8.1.3.2.54 ApduStatus.WarningSonvmcFileFilledUpByTheLastWrite Field

ApduStatus (see page 916) meaning warning processing state of non-volatile memory has changed, file filled up by the last write as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus WarningSonvmcFileFilledUpByTheLastWrite =
```

(ApuStatus)0x6381;

6.2.8.1.3.2.55 ApduStatus.WarningSonvmcNoInformation Field

ApduStatus (see page 916) meaning warning processing state of non-volatile memory has changed, no information given as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus WarningSonvmcNoInformation = (ApuStatus)0x6300;
```

6.2.8.1.3.2.56 ApduStatus.WarningSonvmcXX Field

Mask for ApduStatus (see page 916) meaning warning processing state of non-volatile memory has changed (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ushort WarningSonvmcXX = 0x6300;
```

6.2.8.1.3.2.57 ApduStatus.WarningSonvmuEndOfFileOrRecordBeforeReadingNeBytes Field

ApduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, end of file or record reached before reading Ne bytes as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus WarningSonvmuEndOfFileOrRecordBeforeReadingNeBytes = (ApuStatus)0x6282;
```

6.2.8.1.3.2.58 ApduStatus.WarningSonvmuFileControllInformationIncorrectlyFormatted Field

ApduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, file control information not formatted according to 5.3.3 as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus WarningSonvmuFileControllInformationIncorrectlyFormatted = (ApuStatus)0x6284;
```

6.2.8.1.3.2.59 ApduStatus.WarningSonvmuNoInformation Field

ApduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, no information given as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus WarningSonvmuNoInformation = (ApuStatus)0x6200;
```

6.2.8.1.3.2.60 ApduStatus.WarningSonvmuNoInputDataAvailableFromASensorOnTheCard Field

ApduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, no input data available from a sensor on the card as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus WarningSonvmuNoInputDataAvailableFromASensorOnTheCard = (ApuStatus)0x6286;
```

6.2.8.1.3.2.61 ApduStatus.WarningSonvmuPartOfReturnedDataMayBeCorrupted Field

ApduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, part of returned data may be corrupted as specified in ISO/IEC 7816-4:2005.

C#

```
public static readonly ApduStatus WarningSonvmuPartOfReturnedDataMayBeCorrupted = (ApuStatus)0x6281;
```

6.2.8.1.3.2.62 AduStatus.WarningSonvmuSelectedFileDeactivated Field

AduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, selected file deactivated as specified in ISO/IEC 7816-4:2005.

```
C#  
public static readonly AduStatus WarningSonvmuSelectedFileDeactivated = (AduStatus)0x6283;
```

6.2.8.1.3.2.63 AduStatus.WarningSonvmuSelectedFileInTerminationState Field

AduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, selected file in termination state as specified in ISO/IEC 7816-4:2005.

```
C#  
public static readonly AduStatus WarningSonvmuSelectedFileInTerminationState =  
(AduStatus)0x6285;
```

6.2.8.1.3.2.64 AduStatus.WarningSonvmuTriggeringByTheCardEnd Field

Start of values for AduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

```
C#  
public static readonly ushort WarningSonvmuTriggeringByTheCardEnd = 0x6280;
```

6.2.8.1.3.2.65 AduStatus.WarningSonvmuTriggeringByTheCardStart Field

End of values for AduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged, triggering by the card as specified in ISO/IEC 7816-4:2005.

```
C#  
public static readonly ushort WarningSonvmuTriggeringByTheCardStart = 0x6202;
```

6.2.8.1.3.2.66 AduStatus.WarningSonvmuXX Field

Mask for AduStatus (see page 916) meaning warning processing state of non-volatile memory is unchanged (further qualification in SW2 (see page 941)) as specified in ISO/IEC 7816-4:2005.

```
C#  
public static readonly ushort WarningSonvmuXX = 0x6200;
```

6.2.8.1.3.3 AduStatus Methods

6.2.8.1.3.3.1 AduStatus.Equals Method

Overloaded. Determines whether two Object instances are equal.

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

Parameters

Parameters	Description
object obj	The object to compare with the current object.

Returns

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

6.2.8.1.3.3.2 AduStatus.GetHashCode Method

Serves as a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures

like a hash table.

C#

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

Returns

A hash code for the current Object.

6.2.8.1.3.3.3 ApduStatus.ToString Method

Returns a String that represents the current Object.

C#

```
public override string ToString();
```

Returns

Returns a String that represents the current Object.

6.2.8.1.3.4 ApduStatus Operators

6.2.8.1.3.4.1 ApduStatus.!= Operator

Gets a value indicating whether the ApduStatus ([see page 916](#)) is not identical to a second ApduStatus ([see page 916](#)).

C#

```
public static bool operator !=(ApduStatus v1, ApduStatus v2);
```

Parameters

Parameters	Description
ApduStatus v1	ApduStatus (see page 916) object.
ApduStatus v2	ApduStatus (see page 916) object.

Returns

true if its two operands refer to different objects; otherwise, false.

6.2.8.1.3.4.2 ApduStatus.== Operator

Gets a value indicating whether the ApduStatus ([see page 916](#)) is identical to a second ApduStatus ([see page 916](#)).

C#

```
public static bool operator ==(ApduStatus v1, ApduStatus v2);
```

Parameters

Parameters	Description
ApduStatus v1	ApduStatus (see page 916) object.
ApduStatus v2	ApduStatus (see page 916) object.

Returns

true if the ApduStatus ([see page 916](#)) is identical to a second ApduStatus ([see page 916](#)); otherwiser, false.

6.2.8.1.3.4.3 ApduStatus.usshort Operator

Converts APDU status to ushort.

C#

```
public static explicit operator ushort(ApduStatus value);
```

Parameters

Parameters	Description
ApduStatus value	ApduStatus (see page 916) object.

Returns

ApduStatus (see page 916) converted to byte.

6.2.8.1.3.5 ApduStatus Properties

6.2.8.1.3.5.1 ApduStatus.DataStillAvailableValue Property

Gets the number of available data bytes from the APDU response status.

C#

```
public int DataStillAvailableValue;
```

Property value

The number of available data bytes from the APDU response status.

6.2.8.1.3.5.2 ApduStatus.ErrorSonvmuTriggeringByTheCardValue Property

Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 64XX.

C#

```
public int ErrorSonvmuTriggeringByTheCardValue;
```

Property value

The length of a query that the interface device should recover to try to complete the APDU command .

6.2.8.1.3.5.3 ApduStatus.ErrorWrongLeValue Property

Retrieves the exact number of available data bytes from the APDU response status. Applicable only if APDU status is 6CXX.

C#

```
public int ErrorWrongLeValue;
```

Property value

the exact number of available data bytes in case error specifying wrong APDU response length (Le) field has been received (i.e. SW1SW2 = '6CXX').

6.2.8.1.3.5.4 ApduStatus.IsCheckingError Property

Retrieves value indicating whether APDU response status means checking error.

C#

```
public bool IsCheckingError;
```

Property value

true if checking error occurred; otherwise, false.

6.2.8.1.3.5.5 ApduStatus.IsDataStillAvailable Property

Retrieves a value indicating whether APDU status means normal processing (SW2 (see page 941) encodes the number of data bytes still available).

C#

```
public bool IsDataStillAvailable;
```

Property value

true if number of available data bytes from the APDU response status is available; otherwise, false.

6.2.8.1.3.5.6 ApduStatus.IsErrorCna Property

Retrieves a value indicating whether APDU status means checking error command not allowed (further qualification in SW2 (see page 941)).

C#

```
public bool IsErrorCna;
```

Property value

true if error occurred; otherwise, false.

6.2.8.1.3.5.7 ApduStatus.IsErrorFicns Property

Retrieves a value indicating whether APDU status means checking error functions in CLA not supported (further qualification in SW2 (see page 941)).

C#

```
public bool IsErrorFicns;
```

Property value

true if error occurred; otherwise, false.

6.2.8.1.3.5.8 ApduStatus.IsErrorProprietary67XX Property

Retrieves a value indicating whether APDU status means proprietary error 67XX.

C#

```
public bool IsErrorProprietary67XX;
```

Property value

true if error occurred; otherwise, false.

6.2.8.1.3.5.9 ApduStatus.IsErrorProprietary6BXX Property

Retrieves a value indicating whether APDU status means proprietary error 6BXX.

C#

```
public bool IsErrorProprietary6BXX;
```

Property value

true if error occurred; otherwise, false.

6.2.8.1.3.5.10 ApduStatus.IsErrorProprietary6DXX Property

Retrieves a value indicating whether APDU status means proprietary error 6DXX.

C#

```
public bool IsErrorProprietary6DXX;
```

Property value

true if error occurred; otherwise, false.

6.2.8.1.3.5.11 ApduStatus.IsErrorProprietary6EXX Property

Retrieves a value indicating whether APDU status means proprietary error 6EXX.

C#

```
public bool IsErrorProprietary6EXX;
```

Property value

true if class error occurred; otherwise, false.

6.2.8.1.3.5.12 ApduStatus.IsErrorProprietary6FXX Property

Retrieves a value indicating whether APDU status means proprietary error 6FXX.

C#

```
public bool IsErrorProprietary6FXX;
```

Property value

true if error occurred; otherwise, false.

6.2.8.1.3.5.13 ApduStatus.IsErrorSecurity Property

Retrieves a value indicating whether APDU status means execution error security-related issues.

C#

```
public bool IsErrorSecurity;
```

Property value

true if security error occurred; otherwise, false.

6.2.8.1.3.5.14 ApduStatus.IsErrorSonvmc Property

Retrieves a value indicating whether APDU status means execution error state of nonvolatile memory has changed (further qualification in SW2 (see page 941)).

C#

```
public bool IsErrorSonvmc;
```

Property value

true if state of non-volatile memory has changed; otherwise, false.

6.2.8.1.3.5.15 ApduStatus.IsErrorSonvmu Property

Retrieves a value indicating whether APDU status means execution error state of nonvolatile memory is unchanged (further qualification in SW2 (see page 941)).

C#

```
public bool IsErrorSonvmu;
```

Property value

true if execution error occurred; otherwise, false.

6.2.8.1.3.5.16 ApduStatus.IsErrorSonvmuTriggeredByTheCard Property

Retrieves a value indicating whether APDU status means execution error state of nonvolatile memory is unchanged, triggering by the card.

C#

```
public bool IsErrorSonvmuTriggeringByTheCard;
```

Property value

true if error is triggering by the card; otherwiser, false.

6.2.8.1.3.5.17 ApduStatus.IsErrorWP1P2 Property

Retrieves a value indicating whether APDU status means checking error wrong parameters P1-P2 (further qualification in SW2 (see page 941)).

C#

```
public bool IsErrorWP1P2;
```

Property value

true if error occurred; otherwise, false.

6.2.8.1.3.5.18 ApduStatus.IsErrorWrongLe Property

Retrieves a value indicating whether APDU status means checking error wrong Le field (SW2 (see page 941) encodes the exact number of available data bytes).

C#

```
public bool IsErrorWrongLe;
```

Property value

true if wrong Le field; otherwise, false.

6.2.8.1.3.5.19 ApduStatus.IsExecutionError Property

Retrieves value indicating whether APDU response status means execution error.

C#

```
public bool IsExecutionError;
```

Property value

true if APDU response status (SW1SW2) specifies execution error; otherwise, false.

6.2.8.1.3.5.20 ApduStatus.IsInterindustry Property

Retrieves value indicating whether APDU response status is interindustry.

C#

```
public bool IsInterindustry;
```

Property value

true if APDU response status (SW1SW2) is interindustry; otherwiser, false.

6.2.8.1.3.5.21 ApduStatus.IsNormalProcessing Property

Retrieves value indicating whether APDU response status means normal processing.

C#

```
public bool IsNormalProcessing;
```

Property value

true if APDU response status (SW1SW2) specifies normal processing; otherwiser, false.

6.2.8.1.3.5.22 ApduStatus.IsProcessAborted Property

Retrieves value indicating whether APDU response status means process aborted.

C#

```
public bool IsProcessAborted;
```

Property value

true if APDU processing was aborted; otherwise, false.

6.2.8.1.3.5.23 ApduStatus.IsProcessComplete Property

Retrieves value indicating whether APDU response status means process completed.

C#

```
public bool IsProcessComplete;
```

Property value

true if APDU processing is completed; otherwise, false.

6.2.8.1.3.5.24 ApduStatus.IsProprietary Property

Retrieves value indicating whether APDU response status is proprietary.

C#

```
public bool IsProprietary;
```

Property value

true if APDU response status (SW1SW2) is proprietary; otherwise, false.

6.2.8.1.3.5.25 ApduStatus.IsProprietary9XXX Property

Retrieves a value indicating whether APDU status means proprietary status 9XXX.

C#

```
public bool IsProprietary9XXX;
```

Property value

true if APDU response status (SW1SW2) is proprietary 9XXX; otherwise, false.

6.2.8.1.3.5.26 ApduStatus.IsStateOfNonVolatileMemoryChanged Property

Retrieves value indicating whether APDU response status means state of non-volatile memory has changed.

C#

```
public bool IsStateOfNonVolatileMemoryChanged;
```

Property value

true if state of non-volatile memory has changed; otherwiser, false.

6.2.8.1.3.5.27 ApduStatus.IsStateOfNonVolatileMemoryUnchanged Property

Retrieves value indicating whether APDU response status means state of non-volatile memory has changed state of non-volatile memory is unchanged.

C#

```
public bool IsStateOfNonVolatileMemoryUnchanged;
```

Property value

true if state of non-volatile memory remained unchanged; otherwise, false.

6.2.8.1.3.5.28 ApduStatus.IsValid Property

Retrieves value indicating whether APDU response status is valid, i.e. does not violate any of the SW1SW2 status bytes encoding rules specified in ISO/IEC 7816-3:2006 and ISO/IEC 7816-4:2005.

C#

```
public bool IsValid;
```

Property value

true if APDU command instruction byte (INS) is valid; otherwise, false.

6.2.8.1.3.5.29 ApduStatus.IsWarningProcessing Property

Retrieves value indicating whether APDU response status means warning processing.

C#

```
public bool IsWarningProcessing;
```

Property value

true if APDU response status (SW1SW2) specifies warning processing.

6.2.8.1.3.5.30 ApduStatus.IsWarningSonvmc Property

Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed (further qualification in SW2 (see page 941)).

C#

```
public bool IsWarningSonvmc;
```

Property value

If warning occurred; otherwise, false.

6.2.8.1.3.5.31 ApduStatus.IsWarningSonvmcCounter Property

Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory has changed, counter from 0 to 15 (exact meaning depending on the command).

C#

```
public bool IsWarningSonvmcCounter;
```

Property value

true if counter from 0 to 15 encoded by 'X'; otherwise, false.

6.2.8.1.3.5.32 ApduStatus.IsWarningSonvmu Property

Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged (further qualification in SW2 (see page 941)).

C#

```
public bool IsWarningSonvmu;
```

Property value

true if triggering by the card; otherwise, false.

6.2.8.1.3.5.33 **ApduStatus.IsWarningSonvmuTriggeringByTheCard** Property

Retrieves a value indicating whether APDU status means warning processing state of non-volatile memory is unchanged, triggering by the card.

C#

```
public bool IsWarningSonvmuTriggeringByTheCard;
```

Property value

true if warning is triggering by the card; otherwise, false.

6.2.8.1.3.5.34 **ApduStatus.SW1** Property

Retrieves the first byte of the specified APDU status.

C#

```
public byte SW1;
```

Property value

First status byte.

6.2.8.1.3.5.35 **ApduStatus.SW2** Property

Retrieves the second byte of the specified APDU status.

C#

```
public byte SW2;
```

Property value

Second status byte.

6.2.8.1.3.5.36 **ApduStatus.WarningSonvmcCounterValue** Property

Retrieves the value of a counter returned by the card from the APDU response status. Applicable only if APDU status is 63CX.

C#

```
public int WarningSonvmcCounterValue;
```

Property value

The value of a counter returned by the card (exact meaning depending on the APDU command).

6.2.8.1.3.5.37 **ApduStatus.WarningSonvmuTriggeringByTheCardValue** Property

Retrieves the length of a query that the interface device should recover to try to complete the APDU command that has been previously aborted by the card from the APDU response status. Applicable only if APDU status is 62XX.

C#

```
public int WarningSonvmuTriggeringByTheCardValue;
```

Property value

The length of a further query that the interface device should retrieve before processing the card-originated byte string in the outside world.

6.2.8.1.4 **BerTag** Structure

Represents BER-TLV tag.

C#

```

[Serializable]
public struct BerTag {
    public static readonly BerTag Boolean = (BerTag)1;
    public static readonly BerTag Integer = (BerTag)2;
    public static readonly BerTag OctetString = (BerTag)4;
    public static readonly BerTag Null = (BerTag)5;
    public static readonly BerTag ObjectIdentifier = (BerTag)6;
    public static readonly BerTag ObjectDescriptor = (BerTag)7;
    public static readonly BerTag External = (BerTag)8;
    public static readonly BerTag InstanceOf = (BerTag)8;
    public static readonly BerTag Real = (BerTag)9;
    public static readonly BerTag Enumerated = (BerTag)10;
    public static readonly BerTag EmbeddedPdv = (BerTag)11;
    public static readonly BerTag Utf8String = (BerTag)12;
    public static readonly BerTag RelativeObjectIdentifier = (BerTag)13;
    public static readonly BerTag Sequence = (BerTag)16;
    public static readonly BerTag SequenceOf = (BerTag)16;
    public static readonly BerTag Set = (BerTag)17;
    public static readonly BerTag SetOf = (BerTag)17;
    public static readonly BerTag NumericString = (BerTag)18;
    public static readonly BerTag PrintableString = (BerTag)19;
    public static readonly BerTag TeletexString = (BerTag)20;
    public static readonly BerTag T61String = (BerTag)20;
    public static readonly BerTag VideotexString = (BerTag)21;
    public static readonly BerTag IA5String = (BerTag)22;
    public static readonly BerTag UtcTime = (BerTag)23;
    public static readonly BerTag GeneralizedTime = (BerTag)24;
    public static readonly BerTag GraphicString = (BerTag)25;
    public static readonly BerTag VisibleString = (BerTag)26;
    public static readonly BerTag Iso646String = (BerTag)26;
    public static readonly BerTag GeneralString = (BerTag)27;
    public static readonly BerTag UniversalString = (BerTag)28;
    public static readonly BerTag CharacterString = (BerTag)29;
    public static readonly BerTag BmpString = (BerTag)30;
}

```

BerTag Fields

	Name	Description
	BmpString (see page 945)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying BMPString type.
	Boolean (see page 945)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying boolean type.
	CharacterString (see page 945)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying character string type.
	EmbeddedPdv (see page 945)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying embedded PDV type.
	Enumerated (see page 945)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying enumerated type.
	External (see page 945)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying external type.
	GeneralizedTime (see page 945)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GeneralizedTime type.
	GeneralString (see page 945)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying General-String type.
	GraphicString (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Graphic-String type.
	IA5String (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying IA5String type.
	InstanceOf (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying instance-of type.

	Integer (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying integer type.
	Iso646String (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying ISO646String type.
	Null (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying null type.
	NumericString (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Numeric-String type.
	ObjectDescriptor (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Object-Descriptor type.
	ObjectIdentifier (see page 946)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying object identifier type.
	OctetString (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying octet string type.
	PrintableString (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying PrintableString type.
	Real (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying real type.
	RelativeObjectIdentifier (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying relative object identifier type.
	Sequence (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence type.
	SequenceOf (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence-of type.
	Set (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set type.
	SetOf (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set-of type.
	T61String (see page 947)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying T61String type.
	TeletexString (see page 948)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying TeletexString type.
	UniversalString (see page 948)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Universal-String type.
	UtcTime (see page 948)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTCTime type.
	Utf8String (see page 948)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTF8String type.
	VideotexString (see page 948)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Videotex-String type.
	VisibleString (see page 948)	ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying VisibleString type.

Methods

	Name	Description
	BerTag (see page 944)	Initializes a new instance of the BerTag .

BerTag Methods

	Name	Description
	Equals (see page 948)	Overloaded. Determines whether two Object instances are equal.
	GetHashCode (see page 949)	Serves as a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.
	IsValid (see page 949)	Specifies whether 0xFF value is valid for the first byte of BER-TLV tag fields.

	Read (see page 949)	Reads BER-TLV tag field from memory buffer.
	Save (see page 950)	Saves BER-TLV tag to byte array.
	ToString (see page 950)	Returns a String that represents the current Object.
	Write (see page 950)	Specifies whether 0xFF value is valid for the first byte of BER-TLV tag fields.

BerTag Operators

	Name	Description
	!= (see page 951)	Gets a value indicating whether the BerTag is not identical to a second BerTag.
	== (see page 951)	Gets a value indicating whether the BerTag is identical to a second BerTag.
	int (see page 951)	Converts BER tag to int.

BerTag Properties

	Name	Description
	Class (see page 952)	Gets the class of the BER-TLV tag field.
	Encoding (see page 952)	Gets the encoding of the BER-TLV data object.
	Length (see page 952)	Gets the length of the BER-TLV tag field.
	Number (see page 952)	Gets the tag number encoded in the BERTLV tag field.

6.2.8.1.4.1 BerTag Operator

6.2.8.1.4.1.1 BerTag.BerTag Constructor (BerTagClass, BerTagEncoding, int)

Initializes a new instance of the BerTag .

C#

```
public BerTag(BerTagClass cls, BerTagEncoding encoding, int number);
```

Parameters

Parameters	Description
BerTagClass cls	Class (see page 952) of the BER-TLV tag field.
BerTagEncoding encoding	Encoding (see page 952) of the BER-TLV tag field
int number	Raw tag number of the BER-TLV tag field.

6.2.8.1.4.1.2 BerTag.BerTag Constructor (int)

Initializes a new instance of the BerTag .

C#

```
public BerTag(int value);
```

Parameters

Parameters	Description
int value	BER-TLV tag.

6.2.8.1.4.1.3 BerTag.BerTag Operator (int)

Converts int ([see page 951](#)) to BER tag.

C#

```
public static explicit operator BerTag(int value);
```

Parameters

Parameters	Description
int value	Integer (see page 946) value of BER tag.

Returns

BerTag object.

6.2.8.1.4.2 BerTag Fields**6.2.8.1.4.2.1 BerTag.BmpString Field**

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying BMPString type.

C#

```
public static readonly BerTag BmpString = (BerTag)30;
```

6.2.8.1.4.2.2 BerTag.Boolean Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying boolean type.

C#

```
public static readonly BerTag Boolean = (BerTag)1;
```

6.2.8.1.4.2.3 BerTag.CharacterString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying character string type.

C#

```
public static readonly BerTag CharacterString = (BerTag)29;
```

6.2.8.1.4.2.4 BerTag.EmbeddedPdv Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying embedded PDV type.

C#

```
public static readonly BerTag EmbeddedPdv = (BerTag)11;
```

6.2.8.1.4.2.5 BerTag.Enumerated Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying enumerated type.

C#

```
public static readonly BerTag Enumerated = (BerTag)10;
```

6.2.8.1.4.2.6 BerTag.External Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying external type.

C#

```
public static readonly BerTag External = (BerTag)8;
```

6.2.8.1.4.2.7 BerTag.GeneralizedTime Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying GeneralizedTime type.

C#

```
public static readonly BerTag GeneralizedTime = (BerTag)24;
```

6.2.8.1.4.2.8 BerTag.GeneralString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying General-String type.

C#

```
public static readonly BerTag GeneralString = (BerTag)27;
```

6.2.8.1.4.2.9 BerTag.GraphicString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Graphic-String type.

C#

```
public static readonly BerTag GraphicString = (BerTag)25;
```

6.2.8.1.4.2.10 BerTag.IA5String Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying IA5String type.

C#

```
public static readonly BerTag IA5String = (BerTag)22;
```

6.2.8.1.4.2.11 BerTag.InstanceOf Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying instance-of type.

C#

```
public static readonly BerTag InstanceOf = (BerTag)8;
```

6.2.8.1.4.2.12 BerTag.Integer Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying integer type.

C#

```
public static readonly BerTag Integer = (BerTag)2;
```

6.2.8.1.4.2.13 BerTag.Iso646String Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying ISO646String type.

C#

```
public static readonly BerTag Iso646String = (BerTag)26;
```

6.2.8.1.4.2.14 BerTag.Null Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying null type.

C#

```
public static readonly BerTag Null = (BerTag)5;
```

6.2.8.1.4.2.15 BerTag.NumericString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Numeric-String type.

C#

```
public static readonly BerTag NumericString = (BerTag)18;
```

6.2.8.1.4.2.16 BerTag.ObjectDescriptor Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Object-Descriptor type.

C#

```
public static readonly BerTag ObjectDescriptor = (BerTag)7;
```

6.2.8.1.4.2.17 BerTag.ObjectIdentifier Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying object identifier type.

C#

```
public static readonly BerTag ObjectIdentifier = (BerTag)6;
```

6.2.8.1.4.2.18 BerTag.OctetString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying octet string type.

C#

```
public static readonly BerTag OctetString = (BerTag)4;
```

6.2.8.1.4.2.19 BerTag.PrintableString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying PrintableString type.

C#

```
public static readonly BerTag PrintableString = (BerTag)19;
```

6.2.8.1.4.2.20 BerTag.Real Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying real type.

C#

```
public static readonly BerTag Real = (BerTag)9;
```

6.2.8.1.4.2.21 BerTag.RelativeObjectIdentifier Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying relative object identifier type.

C#

```
public static readonly BerTag RelativeObjectIdentifier = (BerTag)13;
```

6.2.8.1.4.2.22 BerTag.Sequence Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence type.

C#

```
public static readonly BerTag Sequence = (BerTag)16;
```

6.2.8.1.4.2.23 BerTag.SequenceOf Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying sequence-of type.

C#

```
public static readonly BerTag SequenceOf = (BerTag)16;
```

6.2.8.1.4.2.24 BerTag.Set Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set type.

C#

```
public static readonly BerTag Set = (BerTag)17;
```

6.2.8.1.4.2.25 BerTag.SetOf Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying set-of type.

C#

```
public static readonly BerTag SetOf = (BerTag)17;
```

6.2.8.1.4.2.26 BerTag.T61String Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying T61String type.

C#

```
public static readonly BerTag T61String = (BerTag)20;
```

6.2.8.1.4.2.27 BerTag.TeletexString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying TeletexString type.

C#

```
public static readonly BerTag TeletexString = (BerTag)20;
```

6.2.8.1.4.2.28 BerTag.UniversalString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Universal-String type.

C#

```
public static readonly BerTag UniversalString = (BerTag)28;
```

6.2.8.1.4.2.29 BerTag.UtcTime Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTCTime type.

C#

```
public static readonly BerTag UtcTime = (BerTag)23;
```

6.2.8.1.4.2.30 BerTag.Utf8String Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying UTF8String type.

C#

```
public static readonly BerTag Utf8String = (BerTag)12;
```

6.2.8.1.4.2.31 BerTag.VideotexString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying Videotex-String type.

C#

```
public static readonly BerTag VideotexString = (BerTag)21;
```

6.2.8.1.4.2.32 BerTag.VisibleString Field

ASN.1 (ISO/IEC 8825-1:2002) universal class BER-TLV tag specifying VisibleString type.

C#

```
public static readonly BerTag VisibleString = (BerTag)26;
```

6.2.8.1.4.3 BerTag Methods

6.2.8.1.4.3.1 BerTag.Equals Method

Overloaded. Determines whether two Object instances are equal.

C#

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

Parameters

Parameters	Description
object obj	The object to compare with the current object.

Returns

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

6.2.8.1.4.3.2 BerTag.GetHashCode Method

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

C#

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

Returns

A hash code for the current Object.

6.2.8.1.4.3.3 BerTag.IsValid Method

Specifies whether 0xFF value is valid for the first byte of BER-TLV tag fields.

C#

```
public bool IsValid(bool ffIsValidForTheFirstByte);
```

Parameters

Parameters	Description
bool ffIsValidForTheFirstByte	Specifies whether 0xFF value is valid for the first byte of BER-TLV tag fields.

Returns

true if BER-TLV tag field is valid.

6.2.8.1.4.3.4 Read Method

6.2.8.1.4.3.4.1 BerTag.Read Method (IntPtr, int)

Reads BER-TLV tag field from memory buffer.

C#

```
public static BerTag Read(IntPtr buffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer that contains BER-TLV tag field.
int bufferSize	Size of memory buffer that contains BER-TLV tag field.

Returns

BerTag ([see page 941](#)) object.

6.2.8.1.4.3.4.2 BerTag.Read Method (byte[])

Reads BER-TLV tag field from memory buffer.

C#

```
public static BerTag Read(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains BER-TLV tag field.

Returns

BerTag ([see page 941](#)) object.

6.2.8.1.4.3.4.3 BerTag.Read Method (byte[], int, int)

Reads BER-TLV tag field from memory buffer.

C#

```
public static BerTag Read(byte[] buffer, int index, int count);
```

Parameters

Parameters	Description
byte[] buffer	Byte array that contains BER-TLV tag field.
int index	BER-TLV tag field index.
int count	BER-TLV tag fields count.

Returns

BerTag (see page 941) object.

6.2.8.1.4.3.5 BerTag.Save Method

Saves BER-TLV tag to byte array.

C#

```
public byte[] Save();
```

Returns

The size of saved BER-TLV data object.

6.2.8.1.4.3.6 BerTag.ToString Method

Returns a String that represents the current Object.

C#

```
public override string ToString();
```

Returns

Returns a String that represents the current Object.

6.2.8.1.4.3.7 Write Method

6.2.8.1.4.3.7.1 BerTag.Write Method (IntPtr, int)

Specifies whether 0xFF value is valid for the first byte of BER-TLV tag fields.

C#

```
public int Write(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer to write BER-TLV tag field in.
int bufferSize	Size of memory buffer to write BERTLV tag field in.

Returns

The size of written BER-TLV tag field.

6.2.8.1.4.3.7.2 BerTag.Write Method (byte[], int)

Specifies whether 0xFF value is valid for the first byte of BER-TLV tag fields.

C#

```
public int Write(byte[] buffer, int index);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to write BER-TLV tag field in.
int index	BER-TLV tag field index.

Returns

The size of written BER-TLV tag field.

6.2.8.1.4.4 BerTag Operators

6.2.8.1.4.4.1 BerTag.!= Operator

Gets a value indicating whether the BerTag ([see page 941](#)) is not identical to a second BerTag ([see page 941](#)).

C#

```
public static bool operator !=(BerTag v1, BerTag v2);
```

Parameters

Parameters	Description
BerTag v1	BerTag (see page 941) object.
BerTag v2	BerTag (see page 941) object.

Returns

true if its two operands refer to different objects; otherwise, false.

6.2.8.1.4.4.2 BerTag.== Operator

Gets a value indicating whether the BerTag ([see page 941](#)) is identical to a second BerTag ([see page 941](#)).

C#

```
public static bool operator ==(BerTag v1, BerTag v2);
```

Parameters

Parameters	Description
BerTag v1	BerTag (see page 941) object.
BerTag v2	BerTag (see page 941) object.

Returns

true if the BerTag ([see page 941](#)) is identical to a second BerTag ([see page 941](#)); otherwiser, false.

6.2.8.1.4.4.3 BerTag.int Operator

Converts BER tag to int.

C#

```
public static explicit operator int(BerTag value);
```

Parameters

Parameters	Description
BerTag value	BerTag (see page 941) object.

Returns

BerTag (see page 941) converted to int.

6.2.8.1.4.5 BerTag Properties

6.2.8.1.4.5.1 BerTag.Class Property

Gets the class of the BER-TLV tag field.

C#

```
public BerTagClass Class;
```

Property value

BER-TLV tag field Value.

6.2.8.1.4.5.2 BerTag.Encoding Property

Gets the encoding of the BER-TLV data object.

C#

```
public BerTagEncoding Encoding;
```

Property value

BER-TLV tag field Value.

6.2.8.1.4.5.3 BerTag.Length Property

Gets the length of the BER-TLV tag field.

C#

```
public int Length;
```

Property value

BER-TLV tag field value.

6.2.8.1.4.5.4 BerTag.Number Property

Gets the tag number encoded in the BERTLV tag field.

C#

```
public int Number;
```

Property value

BER-TLV tag field value.

6.2.8.1.5 BerTlv Class

Provides functionality for creation, validation, retrieval and update of BER-TLV data objects that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Class Hierarchy

C#

```
public abstract class BerTlv : NObject, ICloneable;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

BerTlv Class

	Name	Description
	Clone (see page 953)	Creates a clone of a BER-TLV data object.
	Create (see page 954)	Creates BER-TLV data object.
	FromHandle (see page 955)	Creates BER-TLV data object from handle.
	FromMemory (see page 955)	Creates BER-TLV data object from memory buffer.
	GetSizeMany (see page 957)	Gets the total size of an array of BER-TLV data objects.
	LoadMany (see page 957)	Creates an array of BER-TLV data objects.
	Save (see page 959)	Writes BER-TLV data object to memory buffer.
	SaveMany (see page 960)	Writes an array of BER-TLV data objects to memory buffer.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

BerTlv Class

	Name	Description
	Length (see page 961)	Gets the BER-TLV data object length field.
	MinLengthSize (see page 961)	Gets the minimal size of the BER-TLV length field.
	Size (see page 961)	Gets the size of a BER-TLV data object.
	Tag (see page 961)	Gets the BER-TLV data object tag field.

6.2.8.1.5.1 BerTlv Methods**6.2.8.1.5.1.1 BerTlv.Clone Method**

Creates a clone of a BER-TLV data object.

C#

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

Returns

The clone of a BerTlv ([see page 952](#)) object.

6.2.8.1.5.1.2 Create Method

6.2.8.1.5.1.2.1 BerTlv.Create Method (BerTag)

Creates BER-TLV data object.

C#

```
public static BerTlv Create(BerTag tag);
```

Parameters

Parameters	Description
BerTag tag	BER-TLV tag field value.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.2.2 BerTlv.Create Method (BerTag, bool)

Creates BER-TLV data object.

C#

```
public static BerTlv Create(BerTag tag, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
BerTag tag	BER-TLV tag field value.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.2.3 BerTlv.Create Method (BerTag, int)

Creates BER-TLV data object.

C#

```
public static BerTlv Create(BerTag tag, int minLengthSize);
```

Parameters

Parameters	Description
BerTag tag	BER-TLV tag field value.
int minLengthSize	Minimal size of the the BER-TLV length field.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.2.4 BerTlv.Create Method (BerTag, int, bool)

Creates BER-TLV data object.

C#

```
public static BerTlv Create(BerTag tag, int minLengthSize, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
BerTag tag	BER-TLV tag field value.
int minLengthSize	Minimal size of the the BER-TLV length field.
bool fflsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.3 FromHandle Method**6.2.8.1.5.1.3.1 BerTlv.FromHandle Method (IntPtr)**

Creates BER-TLV data object from handle.

C#

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

Parameters

Parameters	Description
IntPtr handle	A handle to BER-TLV data object.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.3.2 BerTlv.FromHandle Method (IntPtr, bool)

Creates BER-TLV data object from handle.

C#

```
public static BerTlv FromHandle(IntPtr handle, bool ownsHandle);
```

Parameters

Parameters	Description
IntPtr handle	A handle to BER-TLV data object.
bool ownsHandle	

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.4 FromMemory Method**6.2.8.1.5.1.4.1 BerTlv.FromMemory Method (IntPtr, int)**

Creates BER-TLV data object from memory buffer.

C#

```
public static BerTlv FromMemory(IntPtr buffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer to read BERTLV data from.

int bufferSize	Size (see page 961) of memory buffer to read BER-TLV data from.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.4.2 BerTlv.FromMemory Method (IntPtr, int, bool)

Creates BER-TLV data object from memory buffer.

C#

```
public static BerTlv FromMemory(IntPtr buffer, int bufferSize, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer to read BERTLV data from.
int bufferSize	Size (see page 961) of memory buffer to read BER-TLV data from.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.4.3 BerTlv.FromMemory Method (byte[])

Creates BER-TLV data object from memory buffer.

C#

```
public static BerTlv FromMemory(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLV data from.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.4.4 BerTlv.FromMemory Method (byte[], bool)

Creates BER-TLV data object from memory buffer.

C#

```
public static BerTlv FromMemory(byte[] buffer, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLV data from.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.4.5 BerTlv.FromMemory Method (byte[], int, int)

Creates BER-TLV data object from memory buffer.

C#

```
public static BerTlv FromMemory(byte[] buffer, int index, int count);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLV data from.
int index	BER-TLV tag field index.
int count	BER-TLV tag fields count.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.4.6 BerTlv.FromMemory Method (byte[], int, int, bool)

Creates BER-TLV data object from memory buffer.

C#

```
public static BerTlv FromMemory(byte[] buffer, int index, int count, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLV data from.
int index	BER-TLV tag field index.
int count	BER-TLV tag field count.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

A handle to created BER-TLV data object.

6.2.8.1.5.1.5 BerTlv.GetSizeMany Method

Gets the total size of an array of BER-TLV data objects.

C#

```
public static int GetSizeMany(BerTlv[] berTlvs);
```

Parameters

Parameters	Description
BerTlv[] berTlvs	Array of handles to BER-TLV data objects.

Returns

The total size of all BER-TLV data objects from the array.

6.2.8.1.5.1.6 LoadMany Method

6.2.8.1.5.1.6.1 BerTlv.LoadMany Method (IntPtr, int)

Creates an array of BER-TLV data objects.

C#

```
public static BerTlv[] LoadMany(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer to read BERTLVs data from.
int bufferSize	Size (see page 961) of memory buffer to read BER-TLVs data from.

Returns

Pointer to an array of pointers to handles of created BER-TLV data objects.

6.2.8.1.5.1.6.2 BerTlv.LoadMany Method (IntPtr, int, bool)

Creates an array of BER-TLV data objects.

C#

```
public static BerTlv[] LoadMany(IntPtr pBuffer, int bufferSize, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer to read BERTLVs data from.
int bufferSize	Size (see page 961) of memory buffer to read BER-TLVs data from.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of BER-TLV tag fields.

Returns

Pointer to an array of pointers to handles of created BER-TLV data objects.

6.2.8.1.5.1.6.3 BerTlv.LoadMany Method (byte[])

Creates an array of BER-TLV data objects.

C#

```
public static BerTlv[] LoadMany(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLVs data from.

Returns

Pointer to an array of pointers to handles of created BER-TLV data objects.

6.2.8.1.5.1.6.4 BerTlv.LoadMany Method (byte[], bool)

Creates an array of BER-TLV data objects.

C#

```
public static BerTlv[] LoadMany(byte[] buffer, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLVs data from.

<code>bool ffIsValidForTheTagFirstByte</code>	Boolean parameter specifying whether 0xFF value is valid for the first byte of BER-TLV tag fields.

Returns

Pointer to an array of pointers to handles of created BER-TLV data objects.

6.2.8.1.5.1.6.5 BerTlv.LoadMany Method (byte[], int, int)

Creates an array of BER-TLV data objects.

C#

```
public static BerTlv[] LoadMany(byte[] buffer, int index, int count);
```

Parameters

Parameters	Description
<code>byte[] buffer</code>	Byte array to read BERTLVs data from.
<code>int index</code>	BER-TLV tag field index.
<code>int count</code>	BER-TLV tag fields count.

Returns

Pointer to an array of pointers to handles of created BER-TLV data objects.

6.2.8.1.5.1.6.6 BerTlv.LoadMany Method (byte[], int, int, bool)

Creates an array of BER-TLV data objects.

C#

```
public static BerTlv[] LoadMany(byte[] buffer, int index, int count, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
<code>byte[] buffer</code>	Byte array to read BERTLVs data from.
<code>int index</code>	BER-TLV tag field index.
<code>int count</code>	BER-TLV tag fields count.
<code>bool ffIsValidForTheTagFirstByte</code>	Boolean parameter specifying whether 0xFF value is valid for the first byte of BER-TLV tag fields.

Returns

Pointer to an array of pointers to handles of created BER-TLV data objects.

6.2.8.1.5.1.7 Save Method**6.2.8.1.5.1.7.1 BerTlv.Save Method ()**

Writes BER-TLV data object to memory buffer.

C#

```
public byte[] Save();
```

Returns

Byte array with BER-TLVs.

6.2.8.1.5.1.7.2 BerTlv.Save Method (IntPtr, int)

Writes BER-TLV data object to memory buffer.

C#

```
public int Save(IntPtr buffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer to write BER-TLV in.
int bufferSize	Size (see page 961) of memory buffer to write BER-TLV in.

Returns

Size (see page 961) of written BER-TLV data object.

6.2.8.1.5.1.7.3 BerTlv.Save Method (byte[], int)

Writes BER-TLV data object to memory buffer.

C#

```
public int Save(byte[] buffer, int index);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to write BER-TLV.
int index	BET_TLV index.

Returns

Size (see page 961) of written BER-TLV data object.

6.2.8.1.5.1.8 SaveMany Method**6.2.8.1.5.1.8.1 BerTlv.SaveMany Method (BerTlv[], IntPtr, int)**

Writes an array of BER-TLV data objects to memory buffer.

C#

```
public static int SaveMany(BerTlv[] berTlvs, IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
BerTlv[] berTlvs	An array of handles to BERTLV data objects
IntPtr pBuffer	Pointer to memory buffer to write BER-TLVs in.
int bufferSize	Size (see page 961) of memory buffer to write BER-TLVs in.

Returns

The total size of written BER-TLVs.

6.2.8.1.5.1.8.2 BerTlv.SaveMany Method (BerTlv[], byte[], int)

Writes an array of BER-TLV data objects to memory buffer.

C#

```
public static int SaveMany(BerTlv[] berTlvs, byte[] buffer, int index);
```

Parameters

Parameters	Description
BerTlv[] berTlvs	An array of handles to BERTLV data objects

byte[] buffer	Byte array to write BER-TLVs in.
int index	BER-TLV index.

Returns

The total size of written BER-TLVs.

6.2.8.1.5.2 BerTlv Properties

6.2.8.1.5.2.1 BerTlv.Length Property

Gets the BER-TLV data object length field.

C#

```
public int Length;
```

Property value

The value of the BER-TLV length field.

6.2.8.1.5.2.2 BerTlv.MinLengthSize Property

Gets the minimal size of the BER-TLV length field.

C#

```
public int MinLengthSize;
```

Property value

The minimal size of the BER-TLV length field.

6.2.8.1.5.2.3 BerTlv.Size Property

Gets the size of a BER-TLV data object.

C#

```
public int Size;
```

Property value

The value of the BER-TLV size.

6.2.8.1.5.2.4 BerTlv.Tag Property

Gets the BER-TLV data object tag field.

C#

```
public BerTag Tag;
```

Property value

The value of the BER-TLV tag field.

6.2.8.1.6 ConstructedBerTlv Class

Provides functionality for creation, validation, retrieval and update of constructed BER-TLV data objects that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Class Hierarchy**C#**

```
public sealed class ConstructedBerTlv : BerTlv;
```

ConstructedBerTlv Classes**ConstructedBerTlv Class**

	Name	Description
	DataObjectCollection (see page 963)	Represents the collection of data objects in constructed BER-TLV data object.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

BerTlv Class

	Name	Description
	Clone (see page 953)	Creates a clone of a BER-TLV data object.
	Create (see page 954)	Creates BER-TLV data object.
	FromHandle (see page 955)	Creates BER-TLV data object from handle.
	FromMemory (see page 955)	Creates BER-TLV data object from memory buffer.
	GetSizeMany (see page 957)	Gets the total size of an array of BER-TLV data objects.
	LoadMany (see page 957)	Creates an array of BER-TLV data objects.
	Save (see page 959)	Writes BER-TLV data object to memory buffer.
	SaveMany (see page 960)	Writes an array of BER-TLV data objects to memory buffer.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

BerTlv Class

	Name	Description
	Length (see page 961)	Gets the BER-TLV data object length field.
	MinLengthSize (see page 961)	Gets the minimal size of the BER-TLV length field.
	Size (see page 961)	Gets the size of a BER-TLV data object.
	Tag (see page 961)	Gets the BER-TLV data object tag field.

ConstructedBerTlv Class

	Name	Description
	DataObjects (see page 966)	Gets the BER-TLV data object that is encapsulated in a constructed BER-TLV.

6.2.8.1.6.1 ConstructedBerTlv Classes

6.2.8.1.6.1.1 ConstructedBerTlv.DataObjectCollection Class

Represents the collection of data objects inconstructed BER-TLV data object.

Class Hierarchy

C#

```
public sealed class DataObjectCollection : NObjectCollection<BerTlv>;
```

DataObjectCollection Methods

	Name	Description
	Add ( see page 963)	Adds an item to the ConstructedBerTlv.DataObjectCollection.
	Contains ( see page 966)	Determines whether the specified item is located within the collection.
	IndexOf ( see page 966)	Retrieves the index within the ConstructedBerTlv.DataObjectCollection of the specified integer.

6.2.8.1.6.1.1.1 DataObjectCollection Methods

6.2.8.1.6.1.1.1.1 Add Method

6.2.8.1.6.1.1.1.1.1 ConstructedBerTlv.DataObjectCollection.Add Method (BerTag)

Adds an item to the ConstructedBerTlv.DataObjectCollection ( see page 963).

C#

```
public BerTlv Add(BerTag tag);
```

Parameters

Parameters	Description
BerTag tag	BerTag ( see page 941) object.

Returns

BerTlv ( see page 952) object.

6.2.8.1.6.1.1.1.1.2 ConstructedBerTlv.DataObjectCollection.Add Method (BerTag, bool)

Adds an item to the ConstructedBerTlv.DataObjectCollection ( see page 963).

C#

```
public BerTlv Add(BerTag tag, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
BerTag tag	BerTag ( see page 941) object.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

BerTlv ( see page 952) object.

6.2.8.1.6.1.1.1.1.3 ConstructedBerTlv.DataObjectCollection.Add Method (BerTag, int)

Adds an item to the ConstructedBerTlv.DataObjectCollection ( see page 963).

C#

```
public BerTlv Add(BerTag tag, int minLengthSize);
```

Parameters

Parameters	Description
BerTag tag	BerTag (see page 941) object.
int minLengthSize	Minimal size of the the BER-TLV length field.

Returns

BerTlv (see page 952) object.

6.2.8.1.6.1.1.1.1.4 ConstructedBerTlv.DataObjectCollection.Add Method (BerTag, int, bool)

Adds an item to the ConstructedBerTlv.DataObjectCollection (see page 963).

C#

```
public BerTlv Add(BerTag tag, int minLengthSize, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
BerTag tag	BerTag (see page 941) object.
int minLengthSize	Minimal size of the the BER-TLV length field.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

BerTlv (see page 952) object.

6.2.8.1.6.1.1.1.1.5 ConstructedBerTlv.DataObjectCollection.Add Method (IntPtr, int)

Adds an item to the ConstructedBerTlv.DataObjectCollection (see page 963).

C#

```
public BerTlv Add(IntPtr buffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer to read BERTLV data from.
int bufferSize	Size of memory buffer to read BER-TLV data from.

Returns

BerTlv (see page 952) object.

6.2.8.1.6.1.1.1.1.6 ConstructedBerTlv.DataObjectCollection.Add Method (IntPtr, int, bool)

Adds an item to the ConstructedBerTlv.DataObjectCollection (see page 963).

C#

```
public BerTlv Add(IntPtr buffer, int bufferSize, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer to read BERTLV data from.
int bufferSize	Size of memory buffer to read BER-TLV data from.

bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

BerTlv (see page 952) object.

6.2.8.1.6.1.1.1.1.7 ConstructedBerTlv.DataObjectCollection.Add Method (byte[])

Adds an item to the ConstructedBerTlv.DataObjectCollection (see page 963).

C#

```
public BerTlv Add(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLV data from.

Returns

BerTlv (see page 952) object.

6.2.8.1.6.1.1.1.1.8 ConstructedBerTlv.DataObjectCollection.Add Method (byte[], bool)

Adds an item to the ConstructedBerTlv.DataObjectCollection (see page 963).

C#

```
public BerTlv Add(byte[] buffer, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLV data from.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

BerTlv (see page 952) object.

6.2.8.1.6.1.1.1.1.9 ConstructedBerTlv.DataObjectCollection.Add Method (byte[], int, int)

Adds an item to the ConstructedBerTlv.DataObjectCollection (see page 963).

C#

```
public BerTlv Add(byte[] buffer, int index, int count);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLV data from.
int index	BER-TLV tag field index.
int count	BER-TLV tag field count.

Returns

BerTlv (see page 952) object.

6.2.8.1.6.1.1.1.1.10 ConstructedBerTlv.DataObjectCollection.Add Method (byte[], int, int, bool)

Adds an item to the ConstructedBerTlv.DataObjectCollection (see page 963).

C#

```
public BerTlv Add(byte[] buffer, int index, int count, bool ffIsValidForTheTagFirstByte);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read BERTLV data from.
int index	BER-TLV tag field index.
int count	BER-TLV tag field count.
bool ffIsValidForTheTagFirstByte	Boolean parameter specifying whether 0xFF value is valid for the first byte of the BER-TLV tag field.

Returns

BerTlv ([see page 952](#)) object.

6.2.8.1.6.1.1.1.2 ConstructedBerTlv.DataObjectCollection.Contains Method

Determines whether the specified item is located within the collection.

C#

```
public bool Contains(BerTag tag);
```

Parameters

Parameters	Description
BerTag tag	BerTag (see page 941) object.

Returns

true if the specified object is in the collection; otherwise, false.

6.2.8.1.6.1.1.1.3 ConstructedBerTlv.DataObjectCollection.IndexOf Method

Retrieves the index within the ConstructedBerTlv.DataObjectCollection ([see page 963](#)) of the specified integer.

C#

```
public int IndexOf(BerTag tag);
```

Parameters

Parameters	Description
BerTag tag	BerTag (see page 941) object.

Returns

The zero-based index of the integer in the ConstructedBerTlv.DataObjectCollection ([see page 963](#)).

6.2.8.1.6.2 ConstructedBerTlv Properties**6.2.8.1.6.2.1 ConstructedBerTlv.DataObjects Property**

Gets the BER-TLV data object that is encapsulated in a constructed BER-TLV.

C#

```
public DataObjectCollection DataObjects;
```

Property value

ConstructedBerTlv.DataObjectCollection ([see page 963](#)) object.

6.2.8.1.7 PrimitiveBerTlv Class

Provides functionality for creation, validation, retrieval and update of primitive BER-TLV data objects that are fully compliant with BER-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Class Hierarchy

C#

```
public sealed class PrimitiveBerTlv : BerTlv;
```

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

BerTlv Class

	Name	Description
	Clone (see page 953)	Creates a clone of a BER-TLV data object.
	Create (see page 954)	Creates BER-TLV data object.
	FromHandle (see page 955)	Creates BER-TLV data object from handle.
	FromMemory (see page 955)	Creates BER-TLV data object from memory buffer.
	GetSizeMany (see page 957)	Gets the total size of an array of BER-TLV data objects.
	LoadMany (see page 957)	Creates an array of BER-TLV data objects.
	Save (see page 959)	Writes BER-TLV data object to memory buffer.
	SaveMany (see page 960)	Writes an array of BER-TLV data objects to memory buffer.

PrimitiveBerTlv Class

	Name	Description
	GetValue (see page 968)	Gets the value field of a primitive BER-TLV.
	GetValueAsByte (see page 968)	Retrieves the value field of a primitive BER-TLV as Byte.
	GetValueAsBoolean (see page 969)	Retrieves the value field of a primitive BER-TLV as Boolean.
	GetValueAsInt16 (see page 969)	Retrieves the value field of a primitive BER-TLV as Int16.
	GetValueAsInt32 (see page 969)	Retrieves the value field of a primitive BER-TLV as Int32.
	GetValueAsInteger (see page 969)	Retrieves the value field of a primitive BER-TLV as integer. Can retrieve from 1 to 4 byte value.
	GetValueAsSByte (see page 969)	Retrieves the value field of a primitive BER-TLV as SByte.
	GetValueAsUInt16 (see page 970)	Retrieves the value field of a primitive BER-TLV as UInt16.
	GetValueAsUInt32 (see page 970)	Retrieves the value field of a primitive BER-TLV as UInt32.

	SetLength (see page 970)	Sets the length field of a primitive BER-TLV and updates the size of the BER-TLV respectively.
	SetValue (see page 970)	Sets the value field of a primitive BER-TLV as a Boolean and updates the length field and the size of a BER-TLV respectively.
	SetValueAsInteger (see page 972)	Sets the value field of a primitive BER-TLV as integer. Can set from 1 to 4 byte value.

NObject Properties

NObject Class

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

BerTlv Class

	Name	Description
	Length (see page 961)	Gets the BER-TLV data object length field.
	MinLengthSize (see page 961)	Gets the minimal size of the BER-TLV length field.
	Size (see page 961)	Gets the size of a BER-TLV data object.
	Tag (see page 961)	Gets the BER-TLV data object tag field.

PrimitiveBerTlv Class

	Name	Description
	ValuePtr (see page 972)	Gets the pointer to the value field of a primitive BER-TLV.

6.2.8.1.7.1 PrimitiveBerTlv Methods

6.2.8.1.7.1.1 GetValue Method

6.2.8.1.7.1.1.1 PrimitiveBerTlv.GetValue Method ()

Gets the value field of a primitive BER-TLV.

C#

```
public byte[] GetValue();
```

Returns

Byte array with a primitive BER-TLV.

6.2.8.1.7.1.1.2 PrimitiveBerTlv.GetValue Method (byte[], int)

Gets the value field of a primitive BER-TLV.

C#

```
public void GetValue(byte[] value, int index);
```

Parameters

Parameters	Description
byte[] value	Byte array with the value field of a primitive BER-TLV.
int index	Primitive BER-TLV data object index.

6.2.8.1.7.1.2 PrimitiveBerTlv.GetValueAsByte Method

Retrieves the value field of a primitive BER-TLV as Byte.

C#

```
public byte GetValueAsByte();
```

Returns

The field value of a primitive BER-TLV.

6.2.8.1.7.1.3 PrimitiveBerTlv.GetValueAsBoolean Method

Retrieves the value field of a primitive BER-TLV as Boolean.

C#

```
public bool GetValueAsBoolean();
```

Returns

The field value of a primitive BER-TLV.

6.2.8.1.7.1.4 PrimitiveBerTlv.GetValueAsInt16 Method

Retrieves the value field of a primitive BER-TLV as an Int16.

C#

```
public short GetValueAsInt16();
```

Returns

The field value of a primitive BER-TLV.

6.2.8.1.7.1.5 PrimitiveBerTlv.GetValueAsInt32 Method

Retrieves the value field of a primitive BER-TLV as Int32.

C#

```
public int GetValueAsInt32();
```

Returns

The field value of a primitive BER-TLV.

6.2.8.1.7.1.6 PrimitiveBerTlv.GetValueAsInteger Method

Retrieves the value field of a primitive BER-TLV as integer. Can retrieve from 1 to 4 byte value.

C#

```
public int GetValueAsInteger();
```

Returns

The field value of a primitive BER-TLV.

6.2.8.1.7.1.7 PrimitiveBerTlv.GetValueAsSByte Method

Retrieves the value field of a primitive BER-TLV as SByte.

C#

```
public sbyte GetValueAsSByte();
```

Returns

The field value of a primitive BER-TLV.

6.2.8.1.7.1.8 PrimitiveBerTlv.GetValueAsUInt16 Method

Retrieves the value field of a primitive BER-TLV as UInt16.

C#

```
public ushort GetValueAsUInt16();
```

Returns

The field value of a primitive BER-TLV.

6.2.8.1.7.1.9 PrimitiveBerTlv.GetValueAsUInt32 Method

Retrieves the value field of a primitive BER-TLV as UInt32.

C#

```
public uint GetValueAsUInt32();
```

Returns

The field value of a primitive BER-TLV.

6.2.8.1.7.1.10 PrimitiveBerTlv.SetLength Method

Sets the length field of a primitive BER-TLV and updates the size of the BER-TLV respectively.

C#

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

Parameters

Parameters	Description
int value	BER-TLV length field value.

6.2.8.1.7.1.11 SetValue Method

6.2.8.1.7.1.11.1 PrimitiveBerTlv.SetValue Method (bool)

Sets the value field of a primitive BER-TLV as a Boolean and updates the length field and the size of a BER-TLV respectively.

C#

```
public void SetValue(bool value);
```

Parameters

Parameters	Description
bool value	The primitive BER-TLV value field.

6.2.8.1.7.1.11.2 PrimitiveBerTlv.SetValue Method (byte)

Sets the value field of a primitive BER-TLV as a Byte and updates the length field and the size of a BER-TLV respectively.

C#

```
public void SetValue(byte value);
```

Parameters

Parameters	Description
byte value	The primitive BER-TLV value field.

6.2.8.1.7.1.11.3 PrimitiveBerTlv.SetValue Method (byte[])

Sets the value field of a primitive BER-TLV as a byte array and updates the length field and the size of a BER-TLV respectively.

C#

```
public void SetValue(byte[] value);
```

Parameters

Parameters	Description
byte[] value	Byte array with the value field of a primitive BER-TLV.

6.2.8.1.7.1.11.4 PrimitiveBerTlv.SetValue Method (byte[], int, int)

Sets the value field of a primitive BER-TLV as a part of the byte array and updates the length field and the size of a BER-TLV respectively.

C#

```
public void SetValue(byte[] value, int index, int count);
```

Parameters

Parameters	Description
byte[] value	Byte array with the value field of a primitive BER-TLV.
int index	Primitive BER-TLV index.
int count	Primitive BER-TLV count.

6.2.8.1.7.1.11.5 PrimitiveBerTlv.SetValue Method (int)

Sets the value field of a primitive BER-TLV as a Int32 and updates the length field and the size of a BER-TLV respectively.

C#

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

Parameters

Parameters	Description
int value	The primitive BER-TLV value field.

6.2.8.1.7.1.11.6 PrimitiveBerTlv.SetValue Method (sbyte)

Sets the value field of a primitive BER-TLV as a SByte and updates the length field and the size of a BER-TLV respectively.

C#

```
public void SetValue(sbyte value);
```

Parameters

Parameters	Description
sbyte value	The primitive BER-TLV value field.

6.2.8.1.7.1.11.7 PrimitiveBerTlv.SetValue Method (short)

Sets the value field of a primitive BER-TLV as a Int16 and updates the length field and the size of a BER-TLV respectively.

C#

```
public void SetValue(short value);
```

Parameters

Parameters	Description
short value	The primitive BER-TLV value field.

6.2.8.1.7.1.11.8 PrimitiveBerTlv.SetValue Method (uint)

Sets the value field of a primitive BER-TLV as a UInt32 and updates the length field and the size of a BER-TLV respectively.

C#

```
public void SetValue(uint value);
```

Parameters

Parameters	Description
uint value	The primitive BER-TLV value field.

6.2.8.1.7.1.11.9 PrimitiveBerTlv.SetValue Method (ushort)

Sets the value field of a primitive BER-TLV as a UInt16 and updates the length field and the size of a BER-TLV respectively.

C#

```
public void SetValue(ushort value);
```

Parameters

Parameters	Description
ushort value	The primitive BER-TLV value field.

6.2.8.1.7.1.12 PrimitiveBerTlv.SetValueAsInteger Method

Sets the value field of a primitive BER-TLV as integer. Can set from 1 to 4 byte value.

C#

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

Parameters

Parameters	Description
int value	The field value of a primitive BER-TLV.

6.2.8.1.7.2 PrimitiveBerTlv Properties**6.2.8.1.7.2.1 PrimitiveBerTlv.ValuePtr Property**

Gets the pointer to the value field of a primitive BER-TLV.

C#

```
public IntPtr ValuePtr;
```

Property value

Pointer to the value field of a primitive BER-TLV.

6.2.8.1.8 SCardTags Class

Defines BER-TLV tags that can be used during communication with a smartcard.

Class Hierarchy

C#

```
public static class SCardTags;
```

SCardTags Fields

	Name	Description
	AccountType (see page 977)	Indicates the type of account selected by the cardholder for the transaction (see EMV for coding).
	Address (see page 977)	Address of an individual.
	AnswerToReset (see page 977)	Indicates operating characteristics of the card.
	ApplicationEffectiveDate (see page 977)	Date from which the application can be used, under the responsibility of the application provider.
	ApplicationExpirationDate (see page 978)	Date after which an application expires, under the responsibility of the application provider.
	ApplicationFamilyIdentifier (see page 978)	Application Family Identifier (AFI). Designation of several application areas to enable global interoperability (see ISO/IEC 14443).
	ApplicationIdentifier (see page 978)	Data element identifying an application in the card.
	ApplicationImage (see page 978)	Icon or logo associated with an application.
	ApplicationImageTemplate (see page 978)	Template nesting at least an application image.
	ApplicationLabel (see page 978)	Data element for use at the man-machine interface.
	ApplicationRelatedData (see page 978)	Template nesting parameters of an application.
	ApplicationTemplate (see page 978)	Template identifying an application in the card.
	ApplicationTrack1 (see page 978)	Structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein.
	ApplicationTrack2 (see page 979)	Structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein.
	ApplicationTrack3 (see page 979)	Structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein.
	AuthenticationData (see page 979)	Template nesting authentication data and parameters
	BankIdentifierCode (see page 979)	A universal Bank Identifier Code for use in automated processing in banking and related financial environments (see ISO 9362).
	BiometricDataTemplate (see page 979)	Template nesting biometric reference data objects.
	BiometricInformationGroupTemplate (see page 979)	Template used for nesting biometric information templates
	BiometricInformationTemplate (see page 979)	Template nesting biometric information data objects
	CardCapabilities (see page 979)	Data element fixing card behaviours.
	CardData (see page 980)	Template nesting data related to the card.
	CardEffectiveDate (see page 980)	Date from which the card can be used, under the responsibility of the card issuer.

 SR	CardExpirationDate (see page 980)	Date after which the card expires.
 SR	CardholderBiometricData (see page 980)	Biometric data related to the cardholder.
 SR	CardholderCertificate (see page 980)	Template nesting the cardholder public key, further information, signature of certification authority.
 SR	CardholderHandwrittenSignatureImage (see page 980)	Image of the cardholder's handwritten signature.
 SR	CardholderImageTemplate (see page 980)	Cardholder related images stored within the card.
 SR	CardholderName (see page 980)	Name (see page 987) of the cardholder.
 SR	CardholderNationality (see page 980)	Nationality of the cardholder.
 SR	CardholderPortraitImage (see page 981)	Encoded image data, used for the cardholder portrait image.
 SR	CardholderPrivateKey (see page 981)	Data element containing the cardholder's private key for digital signature functionality using asymmetric mechanisms.
 SR	CardholderPrivateKeyTemplate (see page 981)	Template nesting private key related data objects.
 SR	CardholderPublicKey (see page 981)	Data element containing the cardholder's public key data elements for digital signature functionality using asymmetric mechanisms.
 SR	CardholderPublicKeyTemplate (see page 981)	Template containing the cardholder's public key data objects for digital signature functionality using asymmetric mechanisms.
 SR	CardholderRelatedData (see page 981)	Template nesting data related to the cardholder.
 SR	CardholderRequirementsExcludedFeatures (see page 981)	Data element containing cardholder's requirements for excluded features e.g. cardholder is not able to use fingerprint verification.
 SR	CardholderRequirementsIncludedFeatures (see page 981)	Data element containing a cardholder's requirements for included features e.g. cardholder requires audio assistance from an ATM.
 SR	CardIssuersData (see page 981)	Proprietary.
 SR	CardSequenceNumber (see page 982)	Number distinguishing between separate cards with the same primary account number.
 SR	CardServiceData (see page 982)	Indication of methods available in the card for supporting services.
 SR	CardTrack1 (see page 982)	Structure defined in ISO/IEC 7813[9] and coding in ISO 8583[10], including field separators but excluding start and end sentinels and longitudinal check characters as defined therein. The data content is the same as track 1 of the magnetic stripe, including discretionary data.
 SR	CardTrack2 (see page 982)	Structure defined in ISO/IEC 7813[9] and coding in ISO 8583[10], including field separators but excluding start and end sentinels and longitudinal check characters as defined therein. The data content is the same as track 2 of the magnetic stripe, including discretionary data.
 SR	CardTrack3 (see page 982)	Structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein. The data content is the same as track 3 of the magnetic stripe, including discretionary data.
 SR	CertificateContent (see page 982)	Data element containing the content of a certificate.

	CertificateContentTemplate (see page 982)	Template for nesting certificate content data objects.
	CertificateHolderAuthorization (see page 982)	A certificate holder authorization (e.g. a role identifier) may be contained in a data element or data object with tag.
	CoexistentTagAllocationAuthority (see page 983)	Template used to identify a coexistent tag allocation scheme and the authority responsible for the scheme.
	CommandToPerform (see page 983)	Command APDU.
	CompatibleTagAllocationAuthority (see page 983)	Template used to identify a compatible tag allocation scheme and the authority responsible for the scheme.
	CountryCode (see page 983)	Indication of a country.
	CountryCodeAlpha2Format (see page 983)	Country Code (alpha 2 format). Indication of a country (see ISO 3166-1).
	CountryCodeAlpha3Format (see page 983)	Country Code (alpha 3 format). Indication of a country (see ISO 3166-1).
	CountryCodeAndNationalData (see page 983)	Indication of a country followed by national data and optional national data.
	CurrencyCode (see page 983)	Code for the representation of currencies and funds.
	CurrencyExponent (see page 983)	Number by which an amount of the currency indicated in the card shall be multiplied.
	DateOfBirth (see page 984)	Date of birth of related individual.
	Deprecated (see page 984)	Deprecated in ISO/IEC 7816.
	DigitalSignature (see page 984)	Data element containing a digital signature (asymmetric or symmetric algorithm).
	DigitalSignatureBlock (see page 984)	Template nesting digital signature related data objects.
	DiscretionaryData (see page 984)	Data element not defined in ISO/IEC 7816.
	DiscretionaryDataObjects (see page 984)	Concatenation of data objects not defined in ISO/IEC 7816.
	DisplayControl (see page 984)	Template used to control data displayed at the terminal.
	DisplayMessage (see page 984)	Data element containing a message to display.
	DynamicAuthenticationTemplate (see page 984)	Template used in the command and response data fields of the GENERAL AUTHENTICATE command (defined in ISO/IEC 7816-4).
	DynamicExternalAuthentication (see page 985)	Composite data element used for identifying the algorithm and the key to use in the EXTERNAL AUTHENTICATE command.
	DynamicInternalAuthentication (see page 985)	Composite data element used for identifying the algorithm and the key to use in the INTERNAL AUTHENTICATE command.
	DynamicMutualAuthentication (see page 985)	Composite data element used for identifying the algorithm and the key to use in the mutual authentication process.
	ElementList (see page 985)	Sequence of elements and related information, without identifiers (to be used only within a wrapper).
	ExtendedHeaderList (see page 985)	Data element for indirectly referencing data elements (coding defined in ISO/IEC 7816-4).
	FciTemplate (see page 985)	Template for nesting file control parameters and file management data.
	FcpTemplate (see page 985)	Template for nesting file control parameters.
	FileReference (see page 985)	Reference to a file e.g. a path (coding defined in ISO/IEC 7816-4).
	FmdTemplate (see page 985)	Template for nesting file management data.
	HeaderList (see page 986)	Concatenation of pairs of tag fields and length fields without delimitation (as defined in ISO/IEC 7816-4).
	HistoricalBytes (see page 986)	Used to indicate operating characteristics of the card.

	InitialAccessData (see page 986)	Indication of a command-to-perform for retrieving the initial data string (coding defined in ISO/IEC 7816-4).
	IntegratedCircuitManufacturerIdentifier (see page 986)	Indication of a manufacturer of integrated circuits.
	InterchangeControl (see page 986)	Indication to use in association with a country code to indicate whether international interchange is permitted on a card.
	InterchangeProfile (see page 986)	Data element describing capabilities available in the card to perform an interchange transaction.
	InterindustryDataObjects (see page 986)	Template nesting interindustry data objects.
	InternationalBankAccountNumber (see page 986)	International Bank Account Number (IBAN). A number used internationally to uniquely identify the account of a customer at a financial institution (see ISO 13616).
	IssuerIdentificationNumber (see page 986)	Data element for identifying the card issuer (coding and registration defined in ISO/IEC 7812-1[8]), possibly followed by more data.
	LanguagePreferences (see page 987)	Indication, in order of preference, of up to four languages for the cardholder.
	LoginDataProprietary (see page 987)	Proprietary information intended for connecting the interface device to a remote host, a remote server or an application within these devices.
	LoginTemplate (see page 987)	Template conveying data intended for connecting the interface device to a remote server or an application within such devices (defined in ISO/IEC 7816-4).
	MessageReference (see page 987)	Data element specifying the reference of a message.
	Name (see page 987)	Name of an individual.
	NonInterindustryDataObjects70 (see page 987)	Templates nesting non-interindustry data objects.
	NonInterindustryDataObjects71 (see page 987)	Templates nesting non-interindustry data objects.
	NonInterindustryDataObjects72 (see page 987)	Templates nesting non-interindustry data objects.
	NonInterindustryDataObjects74 (see page 988)	Templates nesting non-interindustry data objects.
	NonInterindustryDataObjects75 (see page 988)	Templates nesting non-interindustry data objects.
	NonInterindustryDataObjects76 (see page 988)	Templates nesting non-interindustry data objects.
	NonInterindustryDataObjects77 (see page 988)	Templates nesting non-interindustry data objects.
	OffsetDataObject (see page 988)	For use with commands using an odd INS code.
	PinUsagePolicy (see page 988)	Indication whether PIN entry is required and under what circumstances.
	PreIssuingData (see page 988)	Proprietary.
	PrimaryAccountNumber (see page 988)	Number identifying a customer account or card.
	PublicKeyOfCertificationAuthority (see page 988)	Data element containing the certification authority's public key for digital signature functionality used to verify certificates.
	QualifiedName (see page 989)	Template nesting the name of an individual and related information, e.g., sex, date of birth etc.
	SecureMessagingTemplate (see page 989)	Template nesting secure messaging data objects.
	SecurityEnvironmentTemplate (see page 989)	Template nesting components of a security environment.

	SecuritySupportTemplate (see page 989)	Template for encapsulating counters and auxiliary data.
	ServiceCode (see page 989)	Identification of geographic / service availability.
	Sex (see page 989)	Gender of an individual.
	SpecialUserRequirements (see page 989)	Template containing at least a tag allocation authority (tag '06', '41', '42' or '4F') and a data object by which this authority indicates the user requirements, possibly related to a disability.
	StaticInternalAuthentication (see page 989)	Data element containing a digital signature value which may be used either alone or in conjunction with the tags StaticInternalAuthenticationFirstAssociatedData (see page 989) and StaticInternalAuthenticationSecondAssociatedData (see page 990).
	StaticInternalAuthenticationFirstAssociatedData (see page 989)	Public key certificate data element to use either alone or in conjunction with the tag StaticInternalAuthenticationSecondAssociatedData (see page 990), to derive a public key value.
	StaticInternalAuthenticationSecondAssociatedData (see page 990)	Data auxiliary to the public key certificate, tag StaticInternalAuthenticationFirstAssociatedData (see page 989), used to derive the notarised public key.
	StatusInformation (see page 990)	Information on card life cycle status and processing status.
	TagList (see page 990)	Concatenation of tag fields without delimitation.
	Timer (see page 990)	Data element specifying the maximal time, in tenths of a second, for performing or executing a process.
	TransactionCounter (see page 990)	Counter incremented under the control of the application in the card after each transaction.
	TransactionDate (see page 990)	Used to recognise the date and time of the last transaction. Length is 4 for YDDD and 10 for full field.
	UniformResourceLocator (see page 990)	Uniform resource locator.
	Wrapper (see page 990)	Template for indirect referencing and retrieval of data elements.

6.2.8.1.8.1 SCardTags Fields

6.2.8.1.8.1.1 SCardTags.AccountType Field

Indicates the type of account selected by the cardholder for the transaction (see EMV for coding).

C#

```
public static readonly BerTag AccountType = (BerTag)0x5F57;
```

6.2.8.1.8.1.2 SCardTags.Address Field

Address of an individual.

C#

```
public static readonly BerTag Address = (BerTag)0x5F42;
```

6.2.8.1.8.1.3 SCardTags.AnswerToReset Field

Indicates operating characteristics of the card.

C#

```
public static readonly BerTag AnswerToReset = (BerTag)0x5F51;
```

6.2.8.1.8.1.4 SCardTags.ApplicationEffectiveDate Field

Date from which the application can be used, under the responsibility of the application provider.

C#

```
public static readonly BerTag ApplicationEffectiveDate = (BerTag)0x5F25;
```

6.2.8.1.8.1.5 SCardTags.ApplicationExpirationDate Field

Date after which an application expires, under the responsibility of the application provider.

C#

```
public static readonly BerTag ApplicationExpirationDate = (BerTag)0x5F24;
```

6.2.8.1.8.1.6 SCardTags.ApplicationFamilyIdentifier Field

Application Family Identifier (AFI). Designation of several application areas to enable global interoperability (see ISO/IEC 14443).

C#

```
public static readonly BerTag ApplicationFamilyIdentifier = (BerTag)0x49;
```

6.2.8.1.8.1.7 SCardTags.ApplicationIdentifier Field

Data element identifying an application in the card.

C#

```
public static readonly BerTag ApplicationIdentifier = (BerTag)0x4F;
```

6.2.8.1.8.1.8 SCardTags.ApplicationImage Field

Icon or logo associated with an application.

C#

```
public static readonly BerTag ApplicationImage = (BerTag)0x5F44;
```

6.2.8.1.8.1.9 SCardTags.ApplicationImageTemplate Field

Template nesting at least an application image.

C#

```
public static readonly BerTag ApplicationImageTemplate = (BerTag)0x6D;
```

6.2.8.1.8.1.10 SCardTags.ApplicationLabel Field

Data element for use at the man-machine interface.

C#

```
public static readonly BerTag ApplicationLabel = (BerTag)0x50;
```

6.2.8.1.8.1.11 SCardTags.ApplicationRelatedData Field

Template nesting parameters of an application.

C#

```
public static readonly BerTag ApplicationRelatedData = (BerTag)0x6E;
```

6.2.8.1.8.1.12 SCardTags.ApplicationTemplate Field

Template identifying an application in the card.

C#

```
public static readonly BerTag ApplicationTemplate = (BerTag)0x61;
```

6.2.8.1.8.1.13 SCardTags.ApplicationTrack1 Field

Structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels

and longitudinal check characters as defined therein.

C#

```
public static readonly BerTag ApplicationTrack1 = (BerTag)0x56;
```

6.2.8.1.8.1.14 SCardTags.ApplicationTrack2 Field

Structure defined in ISO/IEC 7813 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein.

C#

```
public static readonly BerTag ApplicationTrack2 = (BerTag)0x57;
```

6.2.8.1.8.1.15 SCardTags.ApplicationTrack3 Field

Structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein.

C#

```
public static readonly BerTag ApplicationTrack3 = (BerTag)0x58;
```

6.2.8.1.8.1.16 SCardTags.AuthenticationData Field

Template nesting authentication data and parameters

C#

```
public static readonly BerTag AuthenticationData = (BerTag)0x67;
```

6.2.8.1.8.1.17 SCardTags.BankIdentifierCode Field

A universal Bank Identifier Code for use in automated processing in banking and related financial environments (see ISO 9362).

C#

```
public static readonly BerTag BankIdentifierCode = (BerTag)0x5F54;
```

6.2.8.1.8.1.18 SCardTags.BiometricDataTemplate Field

Template nesting biometric reference data objects.

C#

```
public static readonly BerTag BiometricDataTemplate = (BerTag)0x7F2E;
```

6.2.8.1.8.1.19 SCardTags.BiometricInformationGroupTemplate Field

Template used for nesting biometric information templates

C#

```
public static readonly BerTag BiometricInformationGroupTemplate = (BerTag)0x7F61;
```

6.2.8.1.8.1.20 SCardTags.BiometricInformationTemplate Field

Template nesting biometric information data objects

C#

```
public static readonly BerTag BiometricInformationTemplate = (BerTag)0x7F60;
```

6.2.8.1.8.1.21 SCardTags.CardCapabilities Field

Data element fixing card behaviours.

C#

```
public static readonly BerTag CardCapabilities = (BerTag)0x47;
```

6.2.8.1.8.1.22 SCardTags.CardData Field

Template nesting data related to the card.

C#

```
public static readonly BerTag CardData = (BerTag)0x66;
```

6.2.8.1.8.1.23 SCardTags.CardEffectiveDate Field

Date from which the card can be used, under the responsibility of the card issuer.

C#

```
public static readonly BerTag CardEffectiveDate = (BerTag)0x5F26;
```

6.2.8.1.8.1.24 SCardTags.CardExpirationDate Field

Date after which the card expires.

C#

```
public static readonly BerTag CardExpirationDate = (BerTag)0x59;
```

6.2.8.1.8.1.25 SCardTags.CardholderBiometricData Field

Biometric data related to the cardholder.

C#

```
public static readonly BerTag CardholderBiometricData = (BerTag)0x5F2E;
```

6.2.8.1.8.1.26 SCardTags.CardholderCertificate Field

Template nesting the cardholder public key, further information, signature of certification authority.

C#

```
public static readonly BerTag CardholderCertificate = (BerTag)0x7F21;
```

6.2.8.1.8.1.27 SCardTags.CardholderHandwrittenSignatureImage Field

Image of the cardholder's handwritten signature.

C#

```
public static readonly BerTag CardholderHandwrittenSignatureImage = (BerTag)0x5F43;
```

6.2.8.1.8.1.28 SCardTags.CardholderImageTemplate Field

Cardholder related images stored within the card.

C#

```
public static readonly BerTag CardholderImageTemplate = (BerTag)0x6C;
```

6.2.8.1.8.1.29 SCardTags.CardholderName Field

Name (see page 987) of the cardholder.

C#

```
public static readonly BerTag CardholderName = (BerTag)0x5F20;
```

6.2.8.1.8.1.30 SCardTags.CardholderNationality Field

Nationality of the cardholder.

C#

```
public static readonly BerTag CardholderNationality = (BerTag)0x5F2C;
```

6.2.8.1.8.1.31 SCardTags.CardholderPortraitImage Field

Encoded image data, used for the cardholder portrait image.

C#

```
public static readonly BerTag CardholderPortraitImage = (BerTag)0x5F40;
```

6.2.8.1.8.1.32 SCardTags.CardholderPrivateKey Field

Data element containing the cardholder's private key for digital signature functionality using asymmetric mechanisms.

C#

```
public static readonly BerTag CardholderPrivateKey = (BerTag)0x5F48;
```

6.2.8.1.8.1.33 SCardTags.CardholderPrivateKeyTemplate Field

Template nesting private key related data objects.

C#

```
public static readonly BerTag CardholderPrivateKeyTemplate = (BerTag)0x7F48;
```

6.2.8.1.8.1.34 SCardTags.CardholderPublicKey Field

Data element containing the cardholder's public key data elements for digital signature functionality using asymmetric mechanisms.

C#

```
public static readonly BerTag CardholderPublicKey = (BerTag)0x5F49;
```

6.2.8.1.8.1.35 SCardTags.CardholderPublicKeyTemplate Field

Template containing the cardholder's public key data objects for digital signature functionality using asymmetric mechanisms.

C#

```
public static readonly BerTag CardholderPublicKeyTemplate = (BerTag)0x7F49;
```

6.2.8.1.8.1.36 SCardTags.CardholderRelatedData Field

Template nesting data related to the cardholder.

C#

```
public static readonly BerTag CardholderRelatedData = (BerTag)0x65;
```

6.2.8.1.8.1.37 SCardTags.CardholderRequirementsExcludedFeatures Field

Data element containing cardholder's requirements for excluded features e.g. cardholder is not able to use fingerprint verification.

C#

```
public static readonly BerTag CardholderRequirementsExcludedFeatures = (BerTag)0x7F23;
```

6.2.8.1.8.1.38 SCardTags.CardholderRequirementsIncludedFeatures Field

Data element containing a cardholder's requirements for included features e.g. cardholder requires audio assistance from an ATM.

C#

```
public static readonly BerTag CardholderRequirementsIncludedFeatures = (BerTag)0x7F22;
```

6.2.8.1.8.1.39 SCardTags.CardIssuersData Field

Proprietary.

C#

```
public static readonly BerTag CardIssuersData = (BerTag)0x45;
```

6.2.8.1.8.1.40 SCardTags.CardSequenceNumber Field

Number distinguishing between separate cards with the same primary account number.

C#

```
public static readonly BerTag CardSequenceNumber = (BerTag)0x5F34;
```

6.2.8.1.8.1.41 SCardTags.CardServiceData Field

Indication of methods available in the card for supporting services.

C#

```
public static readonly BerTag CardServiceData = (BerTag)0x43;
```

6.2.8.1.8.1.42 SCardTags.CardTrack1 Field

Structure defined in ISO/IEC 7813[9] and coding in ISO 8583[10], including field separators but excluding start and end sentinels and longitudinal check characters as defined therein. The data content is the same as track 1 of the magnetic stripe, including discretionary data.

C#

```
public static readonly BerTag CardTrack1 = (BerTag)0x5F21;
```

6.2.8.1.8.1.43 SCardTags.CardTrack2 Field

Structure defined in ISO/IEC 7813[9] and coding in ISO 8583[10], including field separators but excluding start and end sentinels and longitudinal check characters as defined therein. The data content is the same as track 2 of the magnetic stripe, including discretionary data.

C#

```
public static readonly BerTag CardTrack2 = (BerTag)0x5F22;
```

6.2.8.1.8.1.44 SCardTags.CardTrack3 Field

Structure defined in ISO 4909 and coding in ISO 8583, including field separators but excluding start and end sentinels and longitudinal check characters as defined therein. The data content is the same as track 3 of the magnetic stripe, including discretionary data.

C#

```
public static readonly BerTag CardTrack3 = (BerTag)0x5F23;
```

6.2.8.1.8.1.45 SCardTags.CertificateContent Field

Data element containing the content of a certificate.

C#

```
public static readonly BerTag CertificateContent = (BerTag)0x5F4E;
```

6.2.8.1.8.1.46 SCardTags.CertificateContentTemplate Field

Template for nesting certificate content data objects.

C#

```
public static readonly BerTag CertificateContentTemplate = (BerTag)0x7F4E;
```

6.2.8.1.8.1.47 SCardTags.CertificateHolderAuthorization Field

A certificate holder authorization (e.g. a role identifier) may be contained in a data element or data object with tag.

C#

```
public static readonly BerTag CertificateHolderAuthorization = (BerTag)0x5F4C;
```

6.2.8.1.8.1.48 SCardTags.CoexistentTagAllocationAuthority Field

Template used to identify a coexistent tag allocation scheme and the authority responsible for the scheme.

C#

```
public static readonly BerTag CoexistentTagAllocationAuthority = (BerTag)0x79;
```

6.2.8.1.8.1.49 SCardTags.CommandToPerform Field

Command APDU.

C#

```
public static readonly BerTag CommandToPerform = (BerTag)0x52;
```

6.2.8.1.8.1.50 SCardTags.CompatibleTagAllocationAuthority Field

Template used to identify a compatible tag allocation scheme and the authority responsible for the scheme.

C#

```
public static readonly BerTag CompatibleTagAllocationAuthority = (BerTag)0x78;
```

6.2.8.1.8.1.51 SCardTags.CountryCode Field

Indication of a country.

C#

```
public static readonly BerTag CountryCode = (BerTag)0x5F28;
```

6.2.8.1.8.1.52 SCardTags.CountryCodeAlpha2Format Field

Country Code (alpha 2 format). Indication of a country (see ISO 3166-1).

C#

```
public static readonly BerTag CountryCodeAlpha2Format = (BerTag)0x5F55;
```

6.2.8.1.8.1.53 SCardTags.CountryCodeAlpha3Format Field

Country Code (alpha 3 format). Indication of a country (see ISO 3166-1).

C#

```
public static readonly BerTag CountryCodeAlpha3Format = (BerTag)0x5F56;
```

6.2.8.1.8.1.54 SCardTags.CountryCodeAndNationalData Field

Indication of a country followed by national data and optional national data.

C#

```
public static readonly BerTag CountryCodeAndNationalData = (BerTag)0x41;
```

6.2.8.1.8.1.55 SCardTags.CurrencyCode Field

Code for the representation of currencies and funds.

C#

```
public static readonly BerTag CurrencyCode = (BerTag)0x5F2A;
```

6.2.8.1.8.1.56 SCardTags.CurrencyExponent Field

Number by which an amount of the currency indicated in the card shall be multiplied.

C#

```
public static readonly BerTag CurrencyExponent = (BerTag)0x5F36;
```

6.2.8.1.8.1.57 SCardTags.DateOfBirth Field

Date of birth of related individual.

C#

```
public static readonly BerTag DateOfBirth = (BerTag)0x5F2B;
```

6.2.8.1.8.1.58 SCardTags.Deprecated Field

Deprecated in ISO/IEC 7816.

C#

```
public static readonly BerTag Deprecated = (BerTag)0x5F4B;
```

6.2.8.1.8.1.59 SCardTags.DigitalSignature Field

Data element containing a digital signature (asymmetric or symmetric algorithm).

C#

```
public static readonly BerTag DigitalSignature = (BerTag)0x5F3D;
```

6.2.8.1.8.1.60 SCardTags.DigitalSignatureBlock Field

Template nesting digital signature related data objects.

C#

```
public static readonly BerTag DigitalSignatureBlock = (BerTag)0x7F3D;
```

6.2.8.1.8.1.61 SCardTags.DiscretionaryData Field

Data element not defined in ISO/IEC 7816.

C#

```
public static readonly BerTag DiscretionaryData = (BerTag)0x53;
```

6.2.8.1.8.1.62 SCardTags.DiscretionaryDataObjects Field

Concatenation of data objects not defined in ISO/IEC 7816.

C#

```
public static readonly BerTag DiscretionaryDataObjects = (BerTag)0x73;
```

6.2.8.1.8.1.63 SCardTags.DisplayControl Field

Template used to control data displayed at the terminal.

C#

```
public static readonly BerTag DisplayControl = (BerTag)0x7F20;
```

6.2.8.1.8.1.64 SCardTags.DisplayMessage Field

Data element containing a message to display.

C#

```
public static readonly BerTag DisplayMessage = (BerTag)0x5F45;
```

6.2.8.1.8.1.65 SCardTags.DynamicAuthenticationTemplate Field

Template used in the command and response data fields of the GENERAL AUTHENTICATE command (defined in ISO/IEC 7816-4).

C#

```
public static readonly BerTag DynamicAuthenticationTemplate = (BerTag)0x7C;
```

6.2.8.1.8.1.66 SCardTags.DynamicExternalAuthentication Field

Composite data element used for identifying the algorithm and the key to use in the EXTERNAL AUTHENTICATE command.

C#

```
public static readonly BerTag DynamicExternalAuthentication = (BerTag)0x5F3B;
```

6.2.8.1.8.1.67 SCardTags.DynamicInternalAuthentication Field

Composite data element used for identifying the algorithm and the key to use in the INTERNAL AUTHENTICATE command.

C#

```
public static readonly BerTag DynamicInternalAuthentication = (BerTag)0x5F3A;
```

6.2.8.1.8.1.68 SCardTags.DynamicMutualAuthentication Field

Composite data element used for identifying the algorithm and the key to use in the mutual authentication process.

C#

```
public static readonly BerTag DynamicMutualAuthentication = (BerTag)0x5F3C;
```

6.2.8.1.8.1.69 SCardTags.ElementList Field

Sequence of elements and related information, without identifiers (to be used only within a wrapper).

C#

```
public static readonly BerTag ElementList = (BerTag)0x5F41;
```

6.2.8.1.8.1.70 SCardTags.ExtendedHeaderList Field

Data element for indirectly referencing data elements (coding defined in ISO/IEC 7816-4).

C#

```
public static readonly BerTag ExtendedHeaderList = (BerTag)0x4D;
```

6.2.8.1.8.1.71 SCardTags.FciTemplate Field

Template for nesting file control parameters and file management data.

C#

```
public static readonly BerTag FciTemplate = (BerTag)0x6F;
```

6.2.8.1.8.1.72 SCardTags.FcpTemplate Field

Template for nesting file control parameters.

C#

```
public static readonly BerTag FcpTemplate = (BerTag)0x62;
```

6.2.8.1.8.1.73 SCardTags.FileReference Field

Reference to a file e.g. a path (coding defined in ISO/IEC 7816-4).

C#

```
public static readonly BerTag FileReference = (BerTag)0x51;
```

6.2.8.1.8.1.74 SCardTags.FmdTemplate Field

Template for nesting file management data.

C#

```
public static readonly BerTag FmdTemplate = (BerTag)0x64;
```

6.2.8.1.8.1.75 SCardTags.HeaderList Field

Concatenation of pairs of tag fields and length fields without delimitation (as defined in ISO/IEC 7816-4).

C#

```
public static readonly BerTag HeaderList = (BerTag)0x5D;
```

6.2.8.1.8.1.76 SCardTags.HistoricalBytes Field

Used to indicate operating characteristics of the card.

C#

```
public static readonly BerTag HistoricalBytes = (BerTag)0x5F52;
```

6.2.8.1.8.1.77 SCardTags.InitialAccessData Field

Indication of a command-to-perform for retrieving the initial data string (coding defined in ISO/IEC 7816-4).

C#

```
public static readonly BerTag InitialAccessData = (BerTag)0x44;
```

6.2.8.1.8.1.78 SCardTags.IntegratedCircuitManufacturerIdentifier Field

Indication of a manufacturer of integrated circuits.

C#

```
public static readonly BerTag IntegratedCircuitManufacturerIdentifier = (BerTag)0x5F4D;
```

6.2.8.1.8.1.79 SCardTags.InterchangeControl Field

Indication to use in association with a country code to indicate whether international interchange is permitted on a card.

C#

```
public static readonly BerTag InterchangeControl = (BerTag)0x5F27;
```

6.2.8.1.8.1.80 SCardTags.InterchangeProfile Field

Data element describing capabilities available in the card to perform an interchange transaction.

C#

```
public static readonly BerTag InterchangeProfile = (BerTag)0x5F29;
```

6.2.8.1.8.1.81 SCardTags.InterindustryDataObjects Field

Template nesting interindustry data objects.

C#

```
public static readonly BerTag InterindustryDataObjects = (BerTag)0x7E;
```

6.2.8.1.8.1.82 SCardTags.InternationalBankAccountNumber Field

International Bank Account Number (IBAN). A number used internationally to uniquely identify the account of a customer at a financial institution (see ISO 13616).

C#

```
public static readonly BerTag InternationalBankAccountNumber = (BerTag)0x5F53;
```

6.2.8.1.8.1.83 SCardTags.IssuerIdentificationNumber Field

Data element for identifying the card issuer (coding and registration defined in ISO/IEC 7812-1[8]), possibly followed by more

data.

C#

```
public static readonly BerTag IssuerIdentificationNumber = (BerTag)0x42;
```

6.2.8.1.8.1.84 SCardTags.LanguagePreferences Field

Indication, in order of preference, of up to four languages for the cardholder.

C#

```
public static readonly BerTag LanguagePreferences = (BerTag)0x5F2D;
```

6.2.8.1.8.1.85 SCardTags.LoginDataProprietary Field

Proprietary information intended for connecting the interface device to a remote host, a remote server or an application within these devices.

C#

```
public static readonly BerTag LoginDataProprietary = (BerTag)0x5E;
```

6.2.8.1.8.1.86 SCardTags.LoginTemplate Field

Template conveying data intended for connecting the interface device to a remote server or an application within such devices (defined in ISO/IEC 7816-4).

C#

```
public static readonly BerTag LoginTemplate = (BerTag)0x6A;
```

6.2.8.1.8.1.87 SCardTags.MessageReference Field

Data element specifying the reference of a message.

C#

```
public static readonly BerTag MessageReference = (BerTag)0x5F47;
```

6.2.8.1.8.1.88 SCardTags.Name Field

Name of an individual.

C#

```
public static readonly BerTag Name = (BerTag)0x5B;
```

6.2.8.1.8.1.89 SCardTags.NonInterindustryDataObjects70 Field

Templates nesting non-interindustry data objects.

C#

```
public static readonly BerTag NonInterindustryDataObjects70 = (BerTag)0x70;
```

6.2.8.1.8.1.90 SCardTags.NonInterindustryDataObjects71 Field

Templates nesting non-interindustry data objects.

C#

```
public static readonly BerTag NonInterindustryDataObjects71 = (BerTag)0x71;
```

6.2.8.1.8.1.91 SCardTags.NonInterindustryDataObjects72 Field

Templates nesting non-interindustry data objects.

C#

```
public static readonly BerTag NonInterindustryDataObjects72 = (BerTag)0x72;
```

6.2.8.1.8.1.92 SCardTags.NonInterindustryDataObjects74 Field

Templates nesting non-interindustry data objects.

C#

```
public static readonly BerTag NonInterindustryDataObjects74 = (BerTag)0x74;
```

6.2.8.1.8.1.93 SCardTags.NonInterindustryDataObjects75 Field

Templates nesting non-interindustry data objects.

C#

```
public static readonly BerTag NonInterindustryDataObjects75 = (BerTag)0x75;
```

6.2.8.1.8.1.94 SCardTags.NonInterindustryDataObjects76 Field

Templates nesting non-interindustry data objects.

C#

```
public static readonly BerTag NonInterindustryDataObjects76 = (BerTag)0x76;
```

6.2.8.1.8.1.95 SCardTags.NonInterindustryDataObjects77 Field

Templates nesting non-interindustry data objects.

C#

```
public static readonly BerTag NonInterindustryDataObjects77 = (BerTag)0x77;
```

6.2.8.1.8.1.96 SCardTags.OffsetDataObject Field

For use with commands using an odd INS code.

C#

```
public static readonly BerTag OffsetDataObject = (BerTag)0x54;
```

6.2.8.1.8.1.97 SCardTags.PinUsagePolicy Field

Indication whether PIN entry is required and under what circumstances.

C#

```
public static readonly BerTag PinUsagePolicy = (BerTag)0x5F2F;
```

6.2.8.1.8.1.98 SCardTags.PrelIssuingData Field

Proprietary.

C#

```
public static readonly BerTag PreIssuingData = (BerTag)0x46;
```

6.2.8.1.8.1.99 SCardTags.PrimaryAccountNumber Field

Number identifying a customer account or card.

C#

```
public static readonly BerTag PrimaryAccountNumber = (BerTag)0x5A;
```

6.2.8.1.8.1.100 SCardTags.PublicKeyOfCertificationAuthority Field

Data element containing the certification authority's public key for digital signature functionality used to verify certificates.

C#

```
public static readonly BerTag PublicKeyOfCertificationAuthority = (BerTag)0x5F4A;
```

6.2.8.1.8.1.101 SCardTags.QualifiedName Field

Template nesting the name of an individual and related information, e.g., sex, date of birth etc.

C#

```
public static readonly BerTag QualifiedName = (BerTag)0x6B;
```

6.2.8.1.8.1.102 SCardTags.SecureMessagingTemplate Field

Template nesting secure messaging data objects.

C#

```
public static readonly BerTag SecureMessagingTemplate = (BerTag)0x7D;
```

6.2.8.1.8.1.103 SCardTags.SecurityEnvironmentTemplate Field

Template nesting components of a security environment.

C#

```
public static readonly BerTag SecurityEnvironmentTemplate = (BerTag)0x7B;
```

6.2.8.1.8.1.104 SCardTags.SecuritySupportTemplate Field

Template for encapsulating counters and auxiliary data.

C#

```
public static readonly BerTag SecuritySupportTemplate = (BerTag)0x7A;
```

6.2.8.1.8.1.105 SCardTags.ServiceCode Field

Identification of geographic / service availability.

C#

```
public static readonly BerTag ServiceCode = (BerTag)0x5F30;
```

6.2.8.1.8.1.106 SCardTags.Sex Field

Gender of an individual.

C#

```
public static readonly BerTag Sex = (BerTag)0x5F35;
```

6.2.8.1.8.1.107 SCardTags.SpecialUserRequirements Field

Template containing at least a tag allocation authority (tag '06', '41', '42' or '4F') and a data object by which this authority indicates the user requirements, possibly related to a disability.

C#

```
public static readonly BerTag SpecialUserRequirements = (BerTag)0x68;
```

6.2.8.1.8.1.108 SCardTags.StaticInternalAuthentication Field

Data element containing a digital signature value which may be used either alone or in conjunction with the tags StaticInternalAuthenticationFirstAssociatedData (see page 989) and StaticInternalAuthenticationSecondAssociatedData (see page 990).

C#

```
public static readonly BerTag StaticInternalAuthentication = (BerTag)0x5F37;
```

6.2.8.1.8.1.109 SCardTags.StaticInternalAuthenticationFirstAssociatedData Field

Public key certificate data element to use either alone or in conjunction with the tag StaticInternalAuthenticationSecondAssociatedData (see page 990), to derive a public key value.

C#

```
public static readonly BerTag StaticInternalAuthenticationFirstAssociatedData =  
(BerTag)0x5F38;
```

6.2.8.1.8.1.110 SCardTags.StaticInternalAuthenticationSecondAssociatedData Field

Data auxiliary to the public key certificate, tag StaticInternalAuthenticationFirstAssociatedData (see page 989), used to derive the notarised public key.

C#

```
public static readonly BerTag StaticInternalAuthenticationSecondAssociatedData =  
(BerTag)0x5F39;
```

6.2.8.1.8.1.111 SCardTags.StatusInformation Field

Information on card life cycle status and processing status.

C#

```
public static readonly BerTag StatusInformation = (BerTag)0x48;
```

6.2.8.1.8.1.112 SCardTags.TagList Field

Concatenation of tag fields without delimitation.

C#

```
public static readonly BerTag TagList = (BerTag)0x5C;
```

6.2.8.1.8.1.113 SCardTags.Timer Field

Data element specifying the maximal time, in tenths of a second, for performing or executing a process.

C#

```
public static readonly BerTag Timer = (BerTag)0x5F46;
```

6.2.8.1.8.1.114 SCardTags.TransactionCounter Field

Counter incremented under the control of the application in the card after each transaction.

C#

```
public static readonly BerTag TransactionCounter = (BerTag)0x5F32;
```

6.2.8.1.8.1.115 SCardTags.TransactionDate Field

Used to recognise the date and time of the last transaction. Length is 4 for YDDD and 10 for full field.

C#

```
public static readonly BerTag TransactionDate = (BerTag)0x5F33;
```

6.2.8.1.8.1.116 SCardTags.UniformResourceLocator Field

Uniform resource locator.

C#

```
public static readonly BerTag UniformResourceLocator = (BerTag)0x5F50;
```

6.2.8.1.8.1.117 SCardTags.Wrapper Field

Template for indirect referencing and retrieval of data elements.

C#

```
public static readonly BerTag Wrapper = (BerTag)0x63;
```

6.2.8.1.9 SimpleTag Structure

Represents SIMPLE-TLV tag.

C#

```
[Serializable]
public struct SimpleTag {
}
```

Methods

	Name	Description
	SimpleTag (see page 991)	Initializes a new instance of the SimpleTag class.

SimpleTag Methods

	Name	Description
	Equals (see page 992)	Overloaded. Determines whether two Object instances are equal.
	GetHashCode (see page 992)	Serves as a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.
	ToString (see page 992)	Returns a String that represents the current Object.

SimpleTag Operators

	Name	Description
	!= (see page 992)	Gets a value indicating whether the SimpleTag is not identical to a second SimpleTag.
	== (see page 993)	Gets a value indicating whether the SimpleTag is identical to a second SimpleTag.
	byte (see page 993)	Converts SIMPLE-TLV tag to byte.

SimpleTag Properties

	Name	Description
	IsValid (see page 993)	Represents SIMPLE-TLV tag.

6.2.8.1.9.1 SimpleTag Operator

6.2.8.1.9.1.1 SimpleTag.SimpleTag Constructor (byte)

Initializes a new instance of the SimpleTag class.

C#

```
public SimpleTag(byte value);
```

Parameters

Parameters	Description
byte value	SIMPLE-TLV tag.

6.2.8.1.9.1.2 SimpleTag.SimpleTag Operator (byte)

Converts byte ([see page 993](#)) to SIMPLE-TLV tag.

C#

```
public static explicit operator SimpleTag(byte value);
```


Parameters

Parameters	Description
byte value	Byte value of SIMPLE tag.

Returns

SimpleTag object.

6.2.8.1.9.2 SimpleTag Methods

6.2.8.1.9.2.1 SimpleTag.Equals Method

Overloaded. Determines whether two Object instances are equal.

C#

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

Parameters

Parameters	Description
object obj	The object to compare with the current object.

Returns

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

6.2.8.1.9.2.2 SimpleTag.GetHashCode Method

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

C#

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

Returns

A hash code for the current Object.

6.2.8.1.9.2.3 SimpleTag.ToString Method

Returns a String that represents the current Object.

C#

```
public override string ToString();
```

Returns

Returns a String that represents the current Object.

6.2.8.1.9.3 SimpleTag Operators

6.2.8.1.9.3.1 SimpleTag.!= Operator

Gets a value indicating whether the SimpleTag ([see page 991](#)) is not identical to a second SimpleTag ([see page 991](#)).

C#

```
public static bool operator !=(SimpleTag v1, SimpleTag v2);
```

Parameters

Parameters	Description
SimpleTag v1	SimpleTag (see page 991) object.
SimpleTag v2	SimpleTag (see page 991) object.

Returns

true if its two operands refer to different objects; otherwise, false.

6.2.8.1.9.3.2 SimpleTag.== Operator

Gets a value indicating whether the SimpleTag (see page 991) is identical to a second SimpleTag (see page 991).

C#

```
public static bool operator ==(SimpleTag v1, SimpleTag v2);
```

Parameters

Parameters	Description
SimpleTag v1	SimpleTag (see page 991) object.
SimpleTag v2	SimpleTag (see page 991) object.

Returns

true if the SimpleTag (see page 991) is identical to a second SimpleTag (see page 991); otherwiser, false.

6.2.8.1.9.3.3 SimpleTag.byte Operator

Converts SIMPLE-TLV tag to byte.

C#

```
public static explicit operator byte(SimpleTag value);
```

Parameters

Parameters	Description
SimpleTag value	SimpleTag (see page 991) object.

Returns

SimpleTag (see page 991) converted to byte.

6.2.8.1.9.4 SimpleTag Properties**6.2.8.1.9.4.1 SimpleTag.IsValid Property**

Represents SIMPLE-TLV tag.

C#

```
public bool IsValid;
```

6.2.8.1.10 SimpleTlv Class

Provides functionality for creation, validation, retrieval and update of SIMPLE-TLV data objects that are fully compliant with SIMPLE-TLV encoding rules specified in ISO/IEC 7816-4:2005.

Class Hierarchy**C#**

```
public sealed class SimpleTlv : NObject, ICloneable;
```

Methods**SimpleTlv Class**

	Name	Description
	SimpleTlv (see page 995)	Initializes a new instance of the SimpleTlv class.

NDisposable Methods

	Name	Description
	Dispose (see page 616)	Performs tasks associated with freeing, releasing, or resetting unmanaged resources.

NObject Class

	Name	Description
 	CopyParameters (see page 621)	Copies parameters values from one NObject (see page 621) object to another.
 	Free (see page 621)	Releases memory resources used by specified object.
	GetNativeType (see page 622)	Retrieves native type of object.
	GetParameter (see page 622)	Gets value of parameter by parameter id.
	Reset (see page 622)	Resets all NObject (see page 621) parameters to default values.
	SetParameter (see page 623)	Sets value of parameter by parameter id.

SimpleTlv Class

	Name	Description
	Clone (see page 996)	Creates a clone of a SIMPLE-TLV data object.
 	FromHandle (see page 996)	Creates SIMPLE-TLV data object from handle.
 	GetSizeMany (see page 996)	Gets the total size of an array of SIMPLE-TLV data objects.
	GetValue (see page 997)	Gets the value field of a SIMPLE-TLV data object.
 	LoadMany (see page 997)	Creates an array of SIMPLE-TLV data objects from memory buffer.
	Save (see page 998)	Writes SIMPLE-TLV data object.
 	SaveMany (see page 999)	Writes an array of SIMPLE-TLV data objects to memory buffer.
	SetLength (see page 999)	Sets the length field of a SIMPLE-TLV data object and updates the size of SIMPLE-TLV respectively.
	SetValue (see page 1000)	Sets the value field of a SIMPLE-TLV data object and updates the length field and the size of a SIMPLE-TLV respectively.

NObject Properties**NObject Class**

	Name	Description
	Handle (see page 623)	Gets handle to unmanaged NObject (see page 621).
	Owner (see page 623)	Gets owner of the object.

SimpleTlv Class

	Name	Description
	Length (see page 1000)	Gets the SIMPLE-TLV data object length field.
	Size (see page 1000)	Gets the size of a SIMPLE-TLV data object.
	Tag (see page 1000)	Gets the SIMPLE-TLV data object tag field.
	UseThreeByteLengthAlways (see page 1001)	Sets the option specifying whether 3 bytes size length field should always be used in the SIMPLE-TLV.
	ValuePtr (see page 1001)	Gets the pointer to the value field of a SIMPLE-TLV data object.

6.2.8.1.10.1 SimpleTlv Constructor

6.2.8.1.10.1.1 SimpleTlv.SimpleTlv Constructor (IntPtr, int)

Initializes a new instance of the SimpleTlv class.

C#

```
public SimpleTlv(IntPtr buffer, int bufferLength);
```

Parameters

Parameters	Description
IntPtr buffer	Pointer to buffer with SIMPLE-TLV data object.
int bufferLength	Length (see page 1000) of buffer.

6.2.8.1.10.1.2 SimpleTlv.SimpleTlv Constructor (SimpleTag)

Initializes a new instance of the SimpleTlv class.

C#

```
public SimpleTlv(SimpleTag tag);
```

Parameters

Parameters	Description
SimpleTag tag	SimpleTag (see page 991) object.

6.2.8.1.10.1.3 SimpleTlv.SimpleTlv Constructor (SimpleTag, bool)

Initializes a new instance of the SimpleTlv class.

C#

```
public SimpleTlv(SimpleTag tag, bool useThreeByteLengthAlways);
```

Parameters

Parameters	Description
SimpleTag tag	SimpleTag (see page 991) object.
bool useThreeByteLengthAlways	Specifies whether only 3 bytes size length field should always be used in the SIMPLE-TLV.

6.2.8.1.10.1.4 SimpleTlv.SimpleTlv Constructor (byte[])

Initializes a new instance of the SimpleTlv class.

C#

```
public SimpleTlv(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array with SIMPLE-TLV data object.

6.2.8.1.10.1.5 SimpleTlv.SimpleTlv Constructor (byte[], int, int)

Initializes a new instance of the SimpleTlv class.

C#

```
public SimpleTlv(byte[] buffer, int index, int count);
```

Parameters

Parameters	Description
byte[] buffer	Byte array with SIMPLE-TLV data object.

int index	Index of SIMPLE-TLV data object beginning.
int count	Length (see page 1000) of SIMPLE-TLV data object in buffer.

6.2.8.1.10.2 SimpleTlv Methods

6.2.8.1.10.2.1 SimpleTlv.Clone Method

Creates a clone of a SIMPLE-TLV data object.

C#

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

Returns

SIMPLE-TLV data object.

6.2.8.1.10.2.2 FromHandle Method

6.2.8.1.10.2.2.1 SimpleTlv.FromHandle Method (IntPtr)

Creates SIMPLE-TLV data object from handle.

C#

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

Parameters

Parameters	Description
IntPtr handle	Handle (see page 623) to SIMPLE-TLV data object.

Returns

SimpleTlv (see page 993) object.

6.2.8.1.10.2.2.2 SimpleTlv.FromHandle Method (IntPtr, bool)

Creates SIMPLE-TLV data object from handle.

C#

```
public static SimpleTlv FromHandle(IntPtr handle, bool ownsHandle);
```

Parameters

Parameters	Description
IntPtr handle	Handle (see page 623) to unmanaged SIMPLE-TLV data object.
bool ownsHandle	Indicates whether SimpleTlv (see page 993) takes ownership of the handle.

Returns

SimpleTlv (see page 993) object.

6.2.8.1.10.2.3 SimpleTlv.GetSizeMany Method

Gets the total size of an array of SIMPLE-TLV data objects.

C#

```
public static int GetSizeMany(SimpleTlv[] simpleTlvs);
```

Parameters

Parameters	Description
SimpleTlv[] simpleTlvs	Array of handles to SIMPLE-TLV data objects.

Returns

The total size of all SIMPLE-TLV data objects from the array.

6.2.8.1.10.2.4 GetValue Method**6.2.8.1.10.2.4.1 SimpleTlv.GetValue Method ()**

Gets the value field of a SIMPLE-TLV data object.

C#

```
public byte[] GetValue();
```

Returns

The value field of a SIMPLE-TLV.

6.2.8.1.10.2.4.2 SimpleTlv.GetValue Method (byte[], int)

Gets the value field of a SIMPLE-TLV data object.

C#

```
public void GetValue(byte[] value, int index);
```

Parameters

Parameters	Description
byte[] value	The value field of a SIMPLE-TLV.
int index	Value index.

6.2.8.1.10.2.5 LoadMany Method**6.2.8.1.10.2.5.1 SimpleTlv.LoadMany Method (IntPtr, int)**

Creates an array of SIMPLE-TLV data objects from memory buffer.

C#

```
public static SimpleTlv[] LoadMany(IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr pBuffer	Pointer to memory buffer to read SIMPLE-TLVs data from.
int bufferSize	Size (see page 1000) of memory buffer to read SIMPLE-TLVs data from.

Returns

An array of created SIMPLE-TLV data objects.

6.2.8.1.10.2.5.2 SimpleTlv.LoadMany Method (byte[])

Creates an array of SIMPLE-TLV data objects from memory buffer.

C#

```
public static SimpleTlv[] LoadMany(byte[] buffer);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read SIMPLE-TLVs data from.

Returns

An array of created SIMPLE-TLV data objects.

6.2.8.1.10.2.5.3 SimpleTlv.LoadMany Method (byte[], int, int)

Creates an array of SIMPLE-TLV data objects from memory buffer.

C#

```
public static SimpleTlv[] LoadMany(byte[] buffer, int index, int count);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to read SIMPLE-TLVs data from.
int index	Index in buffer at which reading begins.
int count	The number of created SIMPLE-TLV data objects.

Returns

An array of created SIMPLE-TLV data objects.

6.2.8.1.10.2.6 Save Method**6.2.8.1.10.2.6.1 SimpleTlv.Save Method ()**

Writes SIMPLE-TLV data object.

C#

```
public byte[] Save();
```

Returns

Byte array to write SIMPLE-TLVs in.

6.2.8.1.10.2.6.2 SimpleTlv.Save Method (IntPtr, int)

Writes SIMPLE-TLV data object.

C#

```
public int Save(IntPtr buffer, int bufferSize);
```

Parameters

Parameters	Description
IntPtr buffer	Pointer to memory buffer to write SIMPLE-TLVs in.
int bufferSize	Size (see page 1000) of memory buffer to write SIMPLE-TLVs in.

Returns

The size of written SIMPLE-TLV data object.

6.2.8.1.10.2.6.3 SimpleTlv.Save Method (byte[], int)

Writes SIMPLE-TLV data object.

C#

```
public int Save(byte[] buffer, int index);
```

Parameters

Parameters	Description
byte[] buffer	Byte array to write SIMPLE-TLVs in.
int index	The index in buffer at which saving begins.

Returns

The size of written SIMPLE-TLV data object.

6.2.8.1.10.2.7 SaveMany Method**6.2.8.1.10.2.7.1 SimpleTlv.SaveMany Method (SimpleTlv[], IntPtr, int)**

Writes an array of SIMPLE-TLV data objects to memory buffer.

C#

```
public static int SaveMany(SimpleTlv[] simpleTlvs, IntPtr pBuffer, int bufferSize);
```

Parameters

Parameters	Description
SimpleTlv[] simpleTlvs	SIMPLE-TLV array objects.
IntPtr pBuffer	Pointer to memory buffer to write SIMPLE-TLVs in.
int bufferSize	Size (see page 1000) of memory buffer to write SIMPLETLVs in.

Returns

Total size of written SIMPLE-TLVs.

6.2.8.1.10.2.7.2 SimpleTlv.SaveMany Method (SimpleTlv[], byte[], int)

Writes an array of SIMPLE-TLV data objects to memory buffer.

C#

```
public static int SaveMany(SimpleTlv[] simpleTlvs, byte[] buffer, int index);
```

Parameters

Parameters	Description
SimpleTlv[] simpleTlvs	SIMPLE-TLV array objects.
byte[] buffer	Byte array to write SIMPLE-TLVs in.
int index	Index in buffer at which saving begins.

Returns

Total size of written SIMPLE-TLVs.

6.2.8.1.10.2.8 SimpleTlv.SetLength Method

Sets the length field of a SIMPLE-TLV data object and updates the size of SIMPLE-TLV respectively.

C#

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


Parameters

Parameters	Description
int value	SIMPLE-TLV length field value to be set.

6.2.8.1.10.2.9 SetValue Method**6.2.8.1.10.2.9.1 SimpleTlv.SetValue Method (byte[])**

Sets the value field of a SIMPLE-TLV data object and updates the length field and the size of a SIMPLE-TLV respectively.

C#

```
public void SetValue(byte[] value);
```

Parameters

Parameters	Description
byte[] value	Byte array to be set to the SIMPLE-TLV value field.

6.2.8.1.10.2.9.2 SimpleTlv.SetValue Method (byte[], int, int)

Sets the value field of a SIMPLE-TLV data object and updates the length field and the size of a SIMPLE-TLV respectively.

C#

```
public void SetValue(byte[] value, int index, int count);
```

Parameters

Parameters	Description
byte[] value	Byte array to be set to the SIMPLE-TLV value field.
int index	The index in value parameter at which set data begins.
int count	Count of the value to be set to the SIMPLE-TLV value field.

6.2.8.1.10.3 SimpleTlv Properties**6.2.8.1.10.3.1 SimpleTlv.Length Property**

Gets the SIMPLE-TLV data object length field.

C#

```
public int Length;
```

Property value

The value of the SIMPLE-TLV length field.

6.2.8.1.10.3.2 SimpleTlv.Size Property

Gets the size of a SIMPLE-TLV data object.

C#

```
public int Size;
```

Property value

The SIMPLE-TLV size.

6.2.8.1.10.3.3 SimpleTlv.Tag Property

Gets the SIMPLE-TLV data object tag field.

C#

```
public SimpleTag Tag;
```

Property value

The value of the SIMPLE-TLV tag field.

6.2.8.1.10.3.4 SimpleTlv.UseThreeByteLengthAlways Property

Sets the option specifying whether 3 bytes size length field should always be used in the SIMPLE-TLV.

C#

```
public bool UseThreeByteLengthAlways;
```

Property value

true if only 3 bytes size length field should always be used in the SIMPLE-TLV; otherwise, false.

6.2.8.1.10.3.5 SimpleTlv.ValuePtr Property

Gets the pointer to the value field of a SIMPLE-TLV data object.

C#

```
public IntPtr ValuePtr;
```

Property value

Pointer that receives the pointer to the value field of a SIMPLE-TLV.

6.2.8.2 Structs, Records, Enums

6.2.8.2.1 Neurotec.SmartCards.ApduClassSecureMessaging Enumeration

Specifies secure messaging applied to the APDU command-response pairs according to ISO/IEC 7816-4:2005.

C#

```
[Serializable]
public enum ApduClassSecureMessaging {
    None = 0,
    Proprietary = 1,
    HeaderNotProcessed = 2,
    HeaderAuthenticated = 3
}
```

Members

Members	Description
None = 0	No indication or no secure messaging applied.
Proprietary = 1	Proprietary secure messaging format.
HeaderNotProcessed = 2	Secure messaging according to ISO/IEC 7816-4:2005 section 6. APDU command header not processed according to ISO/IEC 7816-4:2005 section 6.2.3.1.
HeaderAuthenticated = 3	Secure messaging according to ISO/IEC 7816-4:2005 section 6. APDU command header authenticated according to ISO/IEC 7816-4:2005 section 6.2.3.1.

6.2.8.2.2 Neurotec.SmartCards.BerTagClass Enumeration

Specifies a class of a BER-TLV data object.

```
C#
[Serializable]
public enum BerTagClass {
    Universal = 0,
    Application = 1,
    ContextSpecific = 2,
    Private = 3
}
```

Members

Members	Description
Universal = 0	Universal class of a BER-TLV data object.
Application = 1	Application class of a BER-TLV data object.
ContextSpecific = 2	Context-specific class of a BER-TLV data object.
Private = 3	Private class of a BER-TLV data object.

6.2.8.2.3 Neurotec.SmartCards.BerTagEncoding Enumeration

Specifies an encoding of a BER-TLV data object.

```
C#
[Serializable]
public enum BerTagEncoding {
    Primitive = 0,
    Constructed = 1
}
```

Members

Members	Description
Primitive = 0	Primitive encoding of a BER-TLV data object.
Constructed = 1	Constructed encoding of a BER-TLV data object.

6.2.9 Neurotec.SmartCards.Biometry Namespace

Contains biometric-specific types for working with smartcards.

Classes

	Name	Description
	SCardBdtCTTags (see page 1003)	Defines BER-TLV tags that can be used in Challenge (CT) inside Biometric Data Template (BDT).
	SCardBdtTags (see page 1003)	Defines BER-TLV tags that can be used in Biometric Data Template (BDT).
	SCardBitBhtTags (see page 1005)	Defines BER-TLV tags that can be used in Biometric Header Template (BHT) inside Biometric Information Template (BIT).
	SCardBitTags (see page 1007)	Defines BER-TLV tags that can be used in Biometric Information Template (BIT).

Module

.NET Reference (see page 606)

Structs, Records, Enums

	Name	Description
	SCardBiometricSubtype (see page 1009)	Specifies the subtype of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardBiometricType (see page 1009)	Specifies the type of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.
	SCardChallengeQualifier (see page 1010)	Specifies challenge qualifiers that may be used in BER-TLV data objects embedded in the ISO/IEC 7816-11:2004 Challenge Template.

6.2.9.1 Classes

6.2.9.1.1 SCardBdtCTTags Class

Defines BER-TLV tags that can be used in Challenge ([see page 1003](#)) Template (CT) inside Biometric Data Template (BDT).

Class Hierarchy**C#**

```
public static class SCardBdtCTTags;
```

SCardBdtCTTags Fields

	Name	Description
	Challenge (see page 1003)	The tag field of the challenge primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Challenge Template.
	ChallengeQualifier (see page 1003)	The tag field of the challenge qualifier primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Challenge (see page 1003) Template.

6.2.9.1.1.1 SCardBdtCTTags Fields

6.2.9.1.1.1.1 SCardBdtCTTags.Challenge Field

The tag field of the challenge primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Challenge Template.

C#

```
public static readonly BerTag Challenge = (BerTag)0x80;
```

6.2.9.1.1.1.2 SCardBdtCTTags.ChallengeQualifier Field

The tag field of the challenge qualifier primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Challenge ([see page 1003](#)) Template.

C#

```
public static readonly BerTag ChallengeQualifier = (BerTag)0x90;
```

6.2.9.1.2 SCardBdtTags Class

Defines BER-TLV tags that can be used in Biometric Data Template (BDT).

Class Hierarchy**C#**

```
public static class SCardBdtTags;
```

SCardBdtTags Fields

	Name	Description
	ChallengeData (see page 1004)	The tag field of the user prompting challenge primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).
	ChallengeTemplate (see page 1004)	The tag field of the user prompting challenge constructed BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).
	ProprietaryBiometricData (see page 1004)	The tag field of the biometric data with proprietary format primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).
	ProprietaryBiometricDataObjects (see page 1004)	The tag field of the biometric data with proprietary format constructed BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).
	StandardBiometricData (see page 1004)	The tag field of the biometric data with standardized format primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).
	StandardBiometricDataObjects (see page 1005)	The tag field of the biometric data with standardized format constructed BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).

6.2.9.1.2.1 SCardBdtTags Fields**6.2.9.1.2.1.1 SCardBdtTags.ChallengeData Field**

The tag field of the user prompting challenge primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).

C#

```
public static readonly BerTag ChallengeData = (BerTag)0x80;
```

6.2.9.1.2.1.2 SCardBdtTags.ChallengeTemplate Field

The tag field of the user prompting challenge constructed BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).

C#

```
public static readonly BerTag ChallengeTemplate = (BerTag)0xA0;
```

6.2.9.1.2.1.3 SCardBdtTags.ProprietaryBiometricData Field

The tag field of the biometric data with proprietary format primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).

C#

```
public static readonly BerTag ProprietaryBiometricData = (BerTag)0x82;
```

6.2.9.1.2.1.4 SCardBdtTags.ProprietaryBiometricDataObjects Field

The tag field of the biometric data with proprietary format constructed BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).

C#

```
public static readonly BerTag ProprietaryBiometricDataObjects = (BerTag)0xA2;
```

6.2.9.1.2.1.5 SCardBdtTags.StandardBiometricData Field

The tag field of the biometric data with standardized format primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).

C#

```
public static readonly BerTag StandardBiometricData = (BerTag)0x81;
```

6.2.9.1.2.1.6 SCardBdtTags.StandardBiometricDataObjects Field

The tag field of the biometric data with standardized format constructed BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT).

C#

```
public static readonly BerTag StandardBiometricDataObjects = (BerTag)0xA1;
```

6.2.9.1.3 SCardBitBhtTags Class

Defines BER-TLV tags that can be used in Biometric Header Template (BHT) inside Biometric Information Template (BIT).

Class Hierarchy

C#

```
public static class SCardBitBhtTags;
```

SCardBitBhtTags Fields

	Name	Description
	BiometricAlgorithmParametersData (see page 1006)	The tag field of the biometric matching algorithm parameters primitive BER-TLV data object used in the Biometric Header Template (BHT).
	BiometricAlgorithmParametersTemplate (see page 1006)	The tag field of the biometric matching algorithm parameters constructed BER-TLV data object used in the Biometric Header Template (BHT).
	BiometricSubtype (see page 1006)	The tag field of the biometric subtype primitive BER-TLV data object used in the Biometric Header Template (BHT). The biometric subtype BER-TLV must be used in common with the biometric type BER-TLV.
	BiometricType (see page 1006)	The tag field of the biometric type primitive BER-TLV data object used in the Biometric Header Template (BHT).
	CreationDate (see page 1006)	The tag field of the creation date primitive BER-TLV data object used in the Biometric Header Template (BHT).
	Creator (see page 1006)	The tag field of the creator primitive BER-TLV data object used in the Biometric Header Template (BHT).
	FormatOwner (see page 1006)	The tag field of the CBEFF format owner primitive BER-TLV data object used in the Biometric Header Template (BHT).
	FormatType (see page 1007)	The tag field of the CBEFF format type primitive BER-TLV data object used in the Biometric Header Template (BHT).
	Index (see page 1007)	The tag field of the unique reference identifier (index) primitive BER-TLV data object used in the Biometric Header Template (BHT).
	PatronHeaderVersion (see page 1007)	The tag field of the patron header version primitive BER-TLV data object used in the Biometric Header Template (BHT).
	ProductIdentifier (see page 1007)	The tag field of the CBEFF Product Identifier (PID) primitive BER-TLV data object used in the Biometric Header Template (BHT).
	ProprietaryBht (see page 1007)	The tag field of the Biometric Header Template (BHT) constructed BER-TLV data object that corresponds to the biometric data with proprietary format in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT). Such BHT is encapsulated in the superior layer BHT of the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

	StandardBht (see page 1007)	The tag field of the Biometric Header Template (BHT) constructed BER-TLV data object that corresponds to the biometric data with standardized format in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT). Such BHT is encapsulated in the superior layer BHT of the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).
	ValidityPeriod (see page 1007)	The tag field of the validity period primitive BER-TLV data object used in the Biometric Header Template (BHT).

6.2.9.1.3.1 SCardBitBhtTags Fields

6.2.9.1.3.1.1 SCardBitBhtTags.BiometricAlgorithmParametersData Field

The tag field of the biometric matching algorithm parameters primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag BiometricAlgorithmParametersData = (BerTag)0x91;
```

6.2.9.1.3.1.2 SCardBitBhtTags.BiometricAlgorithmParametersTemplate Field

The tag field of the biometric matching algorithm parameters constructed BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag BiometricAlgorithmParametersTemplate = (BerTag)0xB1;
```

6.2.9.1.3.1.3 SCardBitBhtTags.BiometricSubtype Field

The tag field of the biometric subtype primitive BER-TLV data object used in the Biometric Header Template (BHT). The biometric subtype BER-TLV must be used in common with the biometric type BER-TLV.

C#

```
public static readonly BerTag BiometricSubtype = (BerTag)0x82;
```

6.2.9.1.3.1.4 SCardBitBhtTags.BiometricType Field

The tag field of the biometric type primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag BiometricType = (BerTag)0x81;
```

6.2.9.1.3.1.5 SCardBitBhtTags.CreationDate Field

The tag field of the creation date primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag CreationDate = (BerTag)0x83;
```

6.2.9.1.3.1.6 SCardBitBhtTags.Creator Field

The tag field of the creator primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag Creator = (BerTag)0x84;
```

6.2.9.1.3.1.7 SCardBitBhtTags.FormatOwner Field

The tag field of the CBEFF format owner primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag FormatOwner = (BerTag)0x87;
```

6.2.9.1.3.1.8 SCardBitBhtTags.FormatType Field

The tag field of the CBEFF format type primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag FormatType = (BerTag)0x88;
```

6.2.9.1.3.1.9 SCardBitBhtTags.Index Field

The tag field of the unique reference identifier (index) primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag Index = (BerTag)0x90;
```

6.2.9.1.3.1.10 SCardBitBhtTags.PatronHeaderVersion Field

The tag field of the patron header version primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag PatronHeaderVersion = (BerTag)0x80;
```

6.2.9.1.3.1.11 SCardBitBhtTags.ProductIdentifier Field

The tag field of the CBEFF Product Identifier (PID) primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag ProductIdentifier = (BerTag)0x86;
```

6.2.9.1.3.1.12 SCardBitBhtTags.ProprietaryBht Field

The tag field of the Biometric Header Template (BHT) constructed BER-TLV data object that corresponds to the biometric data with proprietary format in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT). Such BHT is encapsulated in the superior layer BHT of the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

C#

```
public static readonly BerTag ProprietaryBht = (BerTag)0xA2;
```

6.2.9.1.3.1.13 SCardBitBhtTags.StandardBht Field

The tag field of the Biometric Header Template (BHT) constructed BER-TLV data object that corresponds to the biometric data with standardized format in the ISO/IEC 7816-11:2004 Biometric Data Template (BDT). Such BHT is encapsulated in the superior layer BHT of the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

C#

```
public static readonly BerTag StandardBht = (BerTag)0xA1;
```

6.2.9.1.3.1.14 SCardBitBhtTags.ValidityPeriod Field

The tag field of the validity period primitive BER-TLV data object used in the Biometric Header Template (BHT).

C#

```
public static readonly BerTag ValidityPeriod = (BerTag)0x85;
```

6.2.9.1.4 SCardBitTags Class

Defines BER-TLV tags that can be used in Biometric Information Template (BIT).

Class Hierarchy

C#

```
public static class SCardBitTags;
```

SCardBitTags Fields

	Name	Description
	AlgorithmReference (see page 1008)	The tag field of the algorithm reference primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).
	BiometricHeaderTemplate (see page 1008)	The tag field of the CBEFF (ISO/IEC 19785-1:2006) compliant Biometric Header Template (BHT) constructed BER-TLV used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).
	ReferenceDataQualifier (see page 1008)	The tag field of the reference data qualifier primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).
	StandardDataObjects (see page 1008)	The tag field of the standard data objects constructed BER-TLV used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).
	TagAllocationAuthorityDataObjects (see page 1008)	The tag field of the tag allocation data objects constructed BER-TLV used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

6.2.9.1.4.1 SCardBitTags Fields**6.2.9.1.4.1.1 SCardBitTags.AlgorithmReference Field**

The tag field of the algorithm reference primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

C#

```
public static readonly BerTag AlgorithmReference = (BerTag)0x80;
```

6.2.9.1.4.1.2 SCardBitTags.BiometricHeaderTemplate Field

The tag field of the CBEFF (ISO/IEC 19785-1:2006) compliant Biometric Header Template (BHT) constructed BER-TLV used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

C#

```
public static readonly BerTag BiometricHeaderTemplate = TagAllocationAuthorityDataObjects;
```

6.2.9.1.4.1.3 SCardBitTags.ReferenceDataQualifier Field

The tag field of the reference data qualifier primitive BER-TLV data object used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

C#

```
public static readonly BerTag ReferenceDataQualifier = (BerTag)0x83;
```

6.2.9.1.4.1.4 SCardBitTags.StandardDataObjects Field

The tag field of the standard data objects constructed BER-TLV used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

C#

```
public static readonly BerTag StandardDataObjects = (BerTag)0xA0;
```

6.2.9.1.4.1.5 SCardBitTags.TagAllocationAuthorityDataObjects Field

The tag field of the tag allocation data objects constructed BER-TLV used in the ISO/IEC 7816-11:2004 Biometric Information Template (BIT).

C#

```
public static readonly BerTag TagAllocationAuthorityDataObjects = (BerTag)0xA1;
```

6.2.9.2 Structs, Records, Enums

6.2.9.2.1 Neurotec.SmartCards.Biometry.SCardBiometricSubtype Enumeration

Specifies the subtype of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.

C#

```
[Serializable]
[Flags]
public enum SCardBiometricSubtype {
    None = 0x00,
    Right = 0x01,
    Left = 0x02,
    Thumb = 0x04,
    PointerFinger = 0x08,
    MiddleFinger = 0x0C,
    RingFinger = 0x10,
    LittleFinger = 0x14
}
```

Members

Members	Description
None = 0x00	No information given.
Right = 0x01	Right.
Left = 0x02	Left.
Thumb = 0x04	Thumb.
PointerFinger = 0x08	Pointer finger.
MiddleFinger = 0x0C	Middle finger.
RingFinger = 0x10	Ring finger.
LittleFinger = 0x14	Little finger.

6.2.9.2.2 Neurotec.SmartCards.Biometry.SCardBiometricType Enumeration

Specifies the type of biometric data according to ISO/IEC 7816-11:2004 and ISO/IEC 19785-3:2007 standards.

C#

```
[Flags]
public enum SCardBiometricType {
    None = 0x00,
    MultipleBiometrics = 0x01,
    FacialFeatures = 0x02,
    Voice = 0x04,
    Fingerprint = 0x08,
    Iris = 0x10,
    Retina = 0x20,
    HandGeometry = 0x40,
    SignatureDynamics = 0x80,
    KeystrokeDynamics = 0x0100,
    LipMovement = 0x0200,
    ThermalFaceImage = 0x0400,
    ThermalHandImage = 0x0800,
    Gait = 0x1000,
    BodyOdor = 0x2000,
    Dna = 0x4000,
    EarShape = 0x8000,
    FingerGeometry = 0x010000,
}
```

```
PalmPrint = 0x020000,  
VeinPattern = 0x040000,  
FootPrint = 0x080000  
}
```

Members

Members	Description
None = 0x00	No information given.
MultipleBiometrics = 0x01	Multiple biometrics used.
FacialFeatures = 0x02	Facial features.
Voice = 0x04	Voice.
Fingerprint = 0x08	Fingerprint.
Iris = 0x10	Iris.
Retina = 0x20	Retina.
HandGeometry = 0x40	Hand geometry.
SignatureDynamics = 0x80	Signature dynamics.
KeystrokeDynamics = 0x0100	Keystroke dynamics.
LipMovement = 0x0200	Lip movement.
ThermalFaceImage = 0x0400	Thermal face image.
ThermalHandImage = 0x0800	Thermal hand image.
Gait = 0x1000	Gait.
BodyOdor = 0x2000	Body odor.
Dna = 0x4000	DNA.
EarShape = 0x8000	Ear shape.
FingerGeometry = 0x010000	Finger geometry.
PalmPrint = 0x020000	Palm print.
VeinPattern = 0x040000	Vein pattern.
FootPrint = 0x080000	Foot print.

6.2.9.2.3 Neurotec.SmartCards.Biometry.SCardChallengeQualifier Enumeration

Specifies challenge qualifiers that may be used in BER-TLV data objects embedded in the ISO/IEC 7816-11:2004 Challenge Template.

C#

```
[Serializable]  
public enum SCardChallengeQualifier {  
    Unspecified = 0,  
    Utf8Coding = 1  
}
```

Members

Members	Description
Unspecified = 0	No information given.
Utf8Coding = 1	UTF8 coding.

6.3 Change Log

In this section change log of the Neurotechnology API components is provided. These abbreviations are used:

- **FIX** - Fixed
- **CHN** - Changed
- **UPD** - Updated
- **ADD** - Added
- **REM** - Removed
- **MER** - Merged

6.3.1 C/C++ API Reference

6.3.1.1 NBiometricStandards Library

Version 3.1.4.7

- **FIX:** FMRecordRemoveFingerView and FmrFingerRemoveFingerView sometimes corrupts ViewNumber of other FmrFingerViews.
- **FIX:** FIRRecordRemoveFingerView and FirFingerRemoveFingerView sometimes corrupts ViewNumber of other FirFingerViews.
- **FIX:** IIRRecordRemoveIrisImage and IirIrisRemoveIrisImage sometimes corrupts ImageNumber of other IirIrisImage.
- **FIX:** Minor fixes.

Version 3.1.4.6

- **FIX:** ANFImageBinaryRecordGetPositions returns incorrect values.

Version 3.1.4.5

- **FIX:** IIRRecord read is now more tolerant with BDIF_NON_STRICT_READ flag and without it.

Version 3.1.4.4

- **FIX:**
ANFPImageAsciiBinaryRecordAddPrintPosition/ANFPImageAsciiBinaryRecordInsertPrintPosition/ANFPImageAsciiBinaryRecordSetPrintPosition
argument validation logic.
- **FIX:** ANType10RecordGetFacialFeaturePoint/ANType10RecordGetFacialFeaturePoints not correctly returning BdifFaceFeaturePoint.Code.
- **FIX:** ANType9RecordAddMinutia/ANType9RecordInsertMinutia/ANType9RecordSetMinutia logic.
- **FIX:** ANFImageBinaryRecordAddPosition/ANFImageBinaryRecordInsertPosition put value to incorrect index.
- **FIX:** ANType14RecordClearSegmentationQualityMetrics clears NQM field, not SQM.

Version 3.1.4.3

- **FIX:** ANTemplateClone.
- **FIX:** ANType9RecordGetOfrsMethod returns incorrect method value.
- **FIX:** ANType17RecordSetMakeModelSerialNumber does not set model value.
- **FIX:** ANType17RecordSetGuid crashes.
- **FIX:** ANType10RecordSetPhotoAttributes crashes.
- **FIX:** ANType10RecordSetSubjectFacialAttributes crashes.

- **FIX:** ANType10RecordGetSmtDescription returns N_E_ARGUMENT_NULL (see page 78) incorrectly.
- **FIX:** ANImageAsciiBinaryRecordSetComment returns N_E_FORMAT (see page 80) incorrectly.
- **FIX:** FMRecordCreateFromNFRecord creates two FingerViews.
- **FIX:** FIRecordCreateFromNImage creates two FingerViews.
- **FIX:** IIRRecordCreateFromNImage creates two IrisImages.

Version 3.1.4.2

- **FIX:** FcrFacelImageSetPoseAngleUncertainty sets Pose values.

Version 3.1.4.1

- **UPD:** Add support for finger card templates license.

Version 3.1.4.0

- **UPD:** Non-strict ANTemplate read now converts from non-ASCII to UTF-8 charset in UDF fields.

Version 3.1.3.1

- **FIX:** More correct handling of Type-1 record DOM field.
- **FIX:** More correct handling of Type-14 record PPC field.

Version 3.1.3.0

- **ADD:** BDIF_ALLOW_QUALITY flag for use with PIV-profile FIRecords.
- **FIX:** FIRecord construction from NImage with resolution units pixels per centimeter.
- **FIX:** FMRecordRecordClone, FIRecordClone and IIRRecordClone of records created with BDIF_DO_NOT_CHECK_CBEFF_PRODUCT_ID (see page 35) flag.
- **FIX:** FMRecordCreateFromFMRecord incorrectly copying SizeX, SizeY, ResolutionX and ResolutionY fields.
- **FIX:** Binary images are not inverted during conversion in ANTemplate.
- **UPD:** Detect byte-pixel-count mismatch for binary uncompressed images in ANTemplate.
- **UPD:** Allow PNG and other compressions (except WSQ) for Type-17 records in ANTemplate.
- **UPD:** Tolerate some more bugs during non-strict read in ANTemplate files.
- **FIX:** PPD/PPC fields handling in Type-13 and Type-14 records in ANTemplate.
- **REM:** Max length constraint for SMT description in ANTemplate.
- **REM:** AN_TYPE_10_RECORD_MAX_SMT_DESCRIPTION_LENGTH macro.
- **CHN:** Allow Type-9 record with user format to have RDG field without MIN field and vice versa.
- **ADD:** ANType9RecordHasMinutiaeRidgeCountsIndicator function.
- **FIX:** Handling of vendor CGA in respect to ANTemplate version is more correct now.

Version 3.1.2.2

- **FIX:** Reading of some ANTemplate files.

Version 3.1.2.1

- **FIX:** Minor fixes.

Version 3.1.2.0

- **CHN:** Rename AN_TYPE_1_RECORD_*_TRANSMITTING_RESOLUTION constants to more correct names.
- **CHN:** Make handling of image scanning resolution and image resolution for Type-3 to Type-8 records more correct in respect to NSR and NTR.
- **CHN:** Rename ANR_SKIP_DUPLICATE_FIELDS constant to ANR_MERGE_DUPLICATE_FIELDS and invert its meaning.

- **ADD:** AN_TYPE_1_RECORD_MAX_RESOLUTION_V4 constant.
- **UPD:** ANTemplate can now handle NSR and NTR fields with values up to 999.99 in standard version 4.0 mode.
- **FIX:** FMRecordCreateFromFMRecord not copying SizeX, SizeY, ResolutionX and ResolutionY fields.
- **UPD:** FIRecordCreateFromMemory with BDIF_NON_STRICT_READ flag now sets CaptureDeviceId to zero if it is invalid.
- **FIX:** Minor fixes.

Version 3.1.1.1

- **FIX:** Reading of some invalid files.
- **FIX:** ANRecordSetIdc.
- **FIX:** Minor fixes.

Version 3.1.1.0

- **UPD:** ANTemplate reading now tolerates space around numbers.
- **ADD:** ANR_RECOVER_FROM_BINARY_DATA flag for use when trying to recover during reading of some files with binary data inside text fields.
- **ADD:** ANT_LEAVE_INVALID_RECORDS_UNVALIDATED flag to leave records than can not pass validation unvalidated during read.

Version 3.1.0.0

- **UPD:** FIRecordCreateFromMemory with BDIF_NON_STRICT_READ flag now tries to auto-select ImageAcquisitionLevel if it is zero.
- **REM:** Obsolete defines, functions and members.
- **UPD:** Now uses NIImages (see page 188) library version 2.6.
- **UPD:** Now uses NTemplates (see page 605) library version 3.1.
- **UPD:** Now uses NCore (see page 74) library version 3.1.

Version 3.0.0.3

- **ADD:** ANR_SKIP_DUPLICATE_FIELDS flag.
- **CHN:** Non-strict ANTemplate read now tolerates TOT field absence.
- **CHN:** WSQ-compression detection works in more situations.
- **FIX:** GMT field validation in ANTemplate.
- **FIX:** Repeated fields are now merged in ANTemplate during read (if ANR_SKIP_DUPLICATE_FIELDS flag is not used).
- **FIX:** Memory leak after removal of ANRecord from ANTemplate or ANTemplateFree.
- **FIX:** Added missing ANRecordGetOwner export.

Version 3.0.0.2

- **UPD:** Constants from CBEFF register.
- **FIX:** Neurotechnology CBEFF BDB constants.
- **UPD:** Internal updates.

Version 3.0.0.1

- **FIX:** Addition of records to ANTemplate from NIImage.
- **FIX:** ANTemplateCreateFromFile crash when file does not exist.

Version 3.0.0.0

- Initial release of NBiometricStandards (see page 13) library.
- **ADD:** support for these standards added: ANSI/NIST-ITL 1-2007, ANSI/NIST-ITL 1-2000, ANSI/NIST-ITL 1-1997 and ANSI/NIST-CSL 1-1993.

6.3.1.2 NCore Library

Version 3.1.1.0

- **CHN:** Move NFileAccess and NByteOrder enum definitions from NTypes.h (see page 133) to NStream.h.

Version 3.1.0.4

- **FIX:** Minor fixes.

Version 3.1.0.3

- **FIX:** Minor fixes.

Version 3.1.0.2

- **FIX:** Minor fixes.

Version 3.1.0.1

- **FIX:** NTypeSupportsParameters function name.

Version 3.1.0.0

- **ADD:** HNObject, a handle to base object for other objects and operations (free and parameters) on it.
- **ADD:** HNTType, a handle to type information for an object and operations on it.
- **ADD:** New, more specific error codes.
- **CHN:** N_CALLBACK (see page 122) is now pointer-to-function.
- **REM:** Obsolete defines, functions and members.
- **UPD:** Refactoring and update of headers.

Version 3.0.0.3

- **FIX:** Crash when file open fails on *nix.

Version 3.0.0.2

- **UPD:** String parameter handling in parameter framework.

Version 3.0.0.1

- **FIX:** Minor fixes.

Version 3.0.0.0

- **UPD:** Unicode support made cross-platform.
- **ADD:** System error (errno) intergration to NErrors.
- **ADD:** SSSE4, SSE4.1, SSE4.2, SSE4A, LZCNT and POPCNT detection support to NProcessorInfo.
- **UPD:** NLibraryInfo structure.
- **ADD:** NCoreOnXxx functions for NCore (see page 74) initialization/cleanup when it is a static library (does nothing when it is a dynamic library).

Version 2.4.4.0

- **UPD:** Error framework made cross-platform.

Version 2.4.3.0

- **ADD:** Aligned memory management functions.

Version 2.4.2.1

- **FIX:** Minor fixes.

Version 2.4.2.0

- **ADD:** NPointF (see page 86), NPointD (see page 85), NSizeF (see page 87), NSizeD (see page 87), NRectF (see page 86) and NRectD (see page 86) types.
- **ADD:** NRange type.
- **UPD:** Internal optimizations.

Version 2.4.1.1

- **FIX:** Minor fixes.

Version 2.4.1.0

- **ADD:** NLibraryInfo module.
- **ADD:** NCoreGetInfo (see page 75) function.

Version 2.4.0.0

- **ADD:** Integration with Win32 and COM errors on Windows.
- **ADD:** NStream module.

Version 2.3.1.0

- **ADD:** NProcessorInfo module for CPU identification on Windows.

Version 2.3.0.1

- **FIX:** Memory leak in parameters framework.

Version 2.3.0.0

- **ADD:** HNStream type.
- **ADD:** Stream integration with .NET.
- **UPD:** Exception integration with .NET.

Version 2.2.2.0

- **CHN:** NMemory interface.

Version 2.2.1.0

- **ADD:** More robust error handling on Windows.

Version 2.2.0.0

- **ADD:** Unicode support.

Version 2.1.0.2

- **FIX:** Functions' calling convention on Windows.

Version 2.1.0.1

- **UPD:** Minor updates.

Version 2.1.0.0

- **REM:** Registration error codes.
- **UPD:** Updated to use Microsoft Visual C++ Runtime Library 8.0.

Version 2.0.1.1

- **CHN:** Minor changes.

Version 2.0.1.0

- **ADD:** NParameters module instead of NMetaTypes module for internal infrastructure support.

- **CHN:** Infrastructure optimization for 64-bit support.

Version 2.0.0.0

- **ADD:** A lot of stuff for internal infrastructure support.
- **CHN:** Some changes in internal infrastructure support.

Version 1.0.0.2

- **ADD:** NIndexPair (see page 115) structure.

Version 1.0.0.1

- **FIX:** Minor fixes in headers.

Version 1.0.0.0

- Initial release.

6.3.1.3 NDeviceManager Library

Version 3.1.0.1

- **FIX:** Minor fixes.

Version 3.1.0.0

- **CHN:** IrisCameraCallback, IrisCameraSetCallback renamed accordingly to IrisCameraIrisAcquiredCallback, IrisCameraSetIrisAcquiredCallback.
- **REM:** Obsolete defines, functions and members.
- **UPD:** Now uses NImages (see page 188) library version 2.6.
- **UPD:** Now uses NCore (see page 74) library version 3.1.

Version 3.0.1.1

- **FIX:** Minor fixes for Camera parameters.

Version 3.0.1.0

- **ADD:** New Camera parameters to support IP cameras.

Version 3.0.0.1

- **UPD:** Internal updates.

Version 3.0.0.0

- New version of NDeviceManager (see page 136) released.
- **MER:** Previous libraries CameraMan, FPScannerMan (see page 143), IrisCameraMan were merged into one library.

6.3.1.4 NExtractors Library

Version 3.1.2.1

- **UPD:** Add support for faces extractor base license.

Version 3.1.2.0

- **CHN:** Changed the lower limit of minimum inner iris boundary radius.

Version 3.1.1.1

- **UPD:** Add support for fingers extractor base license.

Version 3.1.1.0

- **UPD:** Better face pose estimation algorithm.

Version 3.1.0.0

- **UPD:** Better face recognition algorithm.
- **REM:** Obsolete defines, functions and members.
- **UPD:** Now uses NImages (see page 188) library version 2.6.
- **UPD:** Now uses NTemplates (see page 605) library version 3.1.
- **UPD:** Now uses NCore (see page 74) library version 3.1.

Version 3.0.1.0

- **FIX:** Fingerprint/palmprint extraction from 1000 dpi images.
- **UPD:** Palmprint generalization algorithm.
- **FIX:** Minor fixes in face templates generalization algorithm.

Version 3.0.0.0

- Initial release of NExtractors library.
- **CHN:** Previous libraries NEEExtractor, NFExtractor and NLEExtractor were merged into one NExtractors library.
- **UPD:** Better irises extraction algorithm.
- **ADD:** NfeMakeSmallTemplate (see page 156) and NfclsTemplateSmall (see page 156) functions.
- **ADD:** Palmprint extraction support.
- **UPD:** Unicode support made cross-platform.

Change log for previous libraries

NEExtractor Library

Version 1.0.0.2

- **FIX:** Minor fixes.

Version 1.0.0.1

- **FIX:** Minor fixes.

Version 1.0.0.0

- Initial release.

NFExtractor Library

Version 2.1.0.2

- **FIX:** Minor fixes in generalization algorithm.

Version 2.1.0.1

- **ADD:** Missing NfeMakeLightTemplate (see page 160) function.

Version 2.1.0.0

- **UPD:** Generalization algorithm.
- **CHN:** Some modifications in the API.

- **UPD:** Re-linked to new NTemplate library.
- **ADD:** NFE_MODE_XXX constants corresponding to modes from VFExtractor.
- **REM:** NFEP_XXX constants for version info.

Version 2.0.2.1

- **ADD:** NfeGetInfo function.

Version 2.0.2.0

- **UPD:** Internal updates.

Version 2.0.1.0

- **UPD:** Internal optimizations.

Version 2.0.0.4

- **FIX:** Minor fixes.

Version 2.0.0.3

- **FIX:** Minor fixes.

Version 2.0.0.2

- **UPD:** Minor updates.

Version 2.0.0.1

- **FIX:** Generalization with small templates.

Version 2.0.0.0

- **UPD:** Better extraction algorithm.
- **UPD:** Better quality of input image control.
- **ADD:** NFEP_USE_QUALITY (see page 168) and NFEP_QUALITY_THRESHOLD (see page 167) constants and parameters for controlling input image quality control.
- **CHN:** Now NfeExtract (see page 155).../NfeExtractUnpacked... functions return zero-sized template if image quality is low.
- **ADD:** NfeGeneralize (see page 159) and NfeGeneralizaUnpacked functions for generalization of N templates.
- **REM:** NfeUpdateTemplate (see page 157)... functions.
- **ADD:** NFEP_TEMPLATE_SIZE (see page 167) constant and parameter and NfeTemplateSize (see page 162) enum controlling template size/matcher reliability tradeoff.
- **ADD:** More NFEP_MODE_... constants enabling optimizations for more fingerprint scanners.
- **ADD:** Unicode support.
- **UPD:** Updated to use Microsoft Visual C++ Runtime Library 8.0.

Version 1.0.2.0

- **ADD:** Extraction of unpacked template capability.
- **ADD:** Update of edited template capability.
- **UPD:** Minor updates.

Version 1.0.1.2

- **FIX:** Minor fixes.

Version 1.0.1.1

- **FIX:** Minor fixes.

Version 1.0.1.0

- **CHN:** Minutiae neighbors extraction logic. Re-enrollment is required for NMatcher to work properly.
- **ADD:** NFEP_EXTRACTED_RIDGE_COUNTS (see page 167) parameter (can now store indexes of minutiae neighbors in template and it is now the default behavior).
- **UPD:** U.are.U scanner mode.

Version 1.0.0.1

- **FIX:** NfeExtract (see page 155) function division by zero with some images.

Version 1.0.0.0

- Initial release.

NLExtractor Library**Version 1.3.0.1**

- **UPD:** Improved liveness detection algorithm.

Version 1.3.0.0

- **UPD:** New pure face detection is almost twice (~1.9 times) faster.

Version 1.2.0.1

- **FIX:** Minor bug fixes.

Version 1.2.0.0

- **ADD:** Liveness detection when extracting from stream.
- **ADD:** Extraction of template containing multiple records of face features (controlled by parameter NLEP_MAX_RECORDS_PER_TEMPLATE (see page 186)).
- **CHN:** NleDetectEyes() replace with NleDetectFacialFeatures (see page 175).
- **CHN:** Added NleEtractionStatus flag to all extraction functions.
- **CHN:** Extraction, compression and generalization functions take and return HNLTemplate(s) (instead of filling the buffer).
- **CHN:** Parameter NLEP_GENERALIZATION_THRESHOLD is now obsolete.

Version 1.1.0.1

- **UPD:** Re-linked to new NTemplate library.
- **REM:** NLEP_XXX constants for version info.

Version 1.1.0.0

- **UPD:** Improved face and eyes detection.
- **UPD:** Improved enrollment from stream (generates more reliable templates).
- **ADD:** NLEP_FACE_QUALITY_THRESHOLD (see page 185) parameter to control quality of face images.

Version 1.0.1.1

- **ADD:** NleGetInfo function.

Version 1.0.0.1

- **FIX:** Memory leaks in reset function.

Version 1.0.0.0

- Initial release.

6.3.1.5 NImages Library**Version 2.6.1.2**

- **FIX:** WSQ and IHead multi-thread issues.

Version 2.6.1.1

- **FIX:** Reading of some JP2 files with alpha channel.

Version 2.6.1.0

- **UPD:** Support 8-bit RGB-palletized JP2 files.

Version 2.6.0.1

- **FIX:** Cross-platform usage issues.

Version 2.6.0.0

- **ADD:** HNMonochromeImage, HNGrayscaleImage and HNRgbImage handle types to clearly indicate which NImage subclass is used.
- **REM:** Obsolete defines, functions and members.
- **UPD:** Now uses NCore (see page 74) library version 3.1.

Version 2.5.0.4

- **FIX:** NImage diagonal flip produces artifact on some images.
- **FIX:** NImage rotate does not "rotate" horizontal and vertical resolutions.

Version 2.5.0.3

- **FIX:** BmpLoadImageFromHBitmap (see page 190) when HBITMAP is not DIBSECTION.

Version 2.5.0.2

- **FIX:** Save to memory, stream or file without extension in JPEG NImageFormat fails.

Version 2.5.0.1

- **FIX:** Empty file created during save if image or license check error occurs.
- **FIX:** Format-neutral image save with .jpl extension should save in Lossless JPEG format.
- **FIX:** Add missing Lossless JPEG part activation info.
- **FIX:** NImagesGetGrayscaleColorWrapperEx (see page 227) function signature.

Version 2.5.0.0

- **ADD:** Lossless JPEG format support.
- **UPD:** Pro and standard versions of the NImages (see page 188) are now the same library.
- **UPD:** Unicode support made cross-platform.

Version 2.4.0.2

- **FIX:** Crash during CMYK JPEG files reading.
- **FIX:** Memory leak during save in JPEG 2000 format.

Version 2.4.0.1

- **UPD:** Internal updates.

Version 2.4.0.0

- **ADD:** PNG format support.
- **ADD:** JPEG2000 format support.
- **CHN:** RGB to grayscale conversion now uses correct formula.

Version 2.3.0.0

- **ADD:** Image rotation, flipping and cropping functionality.

Version 2.2.0.2

- **ADD:** NImagesGetInfo (see page 228) function.

Version 2.2.0.1

- **FIX:** Some TIFF files reading.

Version 2.2.0.0

- **ADD:** I/O with HNStream.
- **FIX:** Some BMP RLE-compressed files reading.

Version 2.1.0.2

- **FIX:** Saving in JPEG format for some images.

Version 2.1.0.1

- **UPD:** Minor updates.

Version 2.1.0.0

- **ADD:** JPEG format support.
- **FIX:** Memory leak when reading WSQ files.

Version 2.0.1.2

- **FIX:** Minor fixes.

Version 2.0.1.1

- **FIX:** Reading of some BMP files.

Version 2.0.1.0

- **ADD:** Unicode support.

Version 2.0.0.5

- **FIX:** Fixed some TIFF files reading.

Version 2.0.0.4

- **FIX:** Fixed some bad-formed BMP files reading.

Version 2.0.0.3

- **UPD:** Updated to use Microsoft Visual C++ Runtime Library 8.0.
- **CHN:** Renamed to NImages (see page 188) to be consistent with other components.

Version 2.0.0.2

- **FIX:** Some BMP files reading.

Version 2.0.0.1

- **CHN:** Minor internal changes.

Version 2.0.0.0

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

Version 1.0.0.2

- **FIX:** Minor fixes in header files.

Version 1.0.0.1

- **FIX:** Minor fixes.

Version 1.0.0.0

- Initial release.

6.3.1.6 NLicensing Library

Version 3.1.0.0

- **ADD:** `NLicenseIsComponentActivated` (see page 262) function.
- **UPD:** Now uses `NCore` (see page 74) library version 3.1.

Version 3.0.0.0

- New version of the `NLicensing` (see page 260) library released.

6.3.1.7 NSmartCards Library

Version 1.2.0.2

- **FIX:** Handling of three-byte BER tags.

Version 1.2.0.1

- **FIX:** `SimpleTlvCreate` (see page 506) does not set tag.

Version 1.2.0.0

- **REM:** Obsolete defines, functions and members.
- **UPD:** Now uses `NCore` (see page 74) library version 3.1.

Version 1.1.0.1

- **FIX:** `ApduStatusIs*` macros.

Version 1.1.0.0**Version 1.0.0.0**

- Initial release.

6.3.1.8 NTemplates Library

Version 3.1.0.2

- **REM:** `NERecordGetMaxSize` dummy function.

Version 3.1.0.1

- **FIX:** Creation of empty palm `NFRecord`.

Version 3.1.0.0

- **ADD:** `NBiometricType` enum.

- **CHN:** NFRecordTruncateMinutiaeByQuality (see page 570) will no longer generate error if template does not contain minutia qualities - will just do nothing.
- **REM:** Obsolete defines, functions and members.
- **UPD:** Now uses NCore (see page 74) library version 3.1.

Version 3.0.0.2

- **UPD:** Minor internal updates.

Version 3.0.0.1

- **FIX:** Palmprint with ridge counts template pack/unpack.

Version 3.0.0.0

- **CHN:** NTemplate library renamed to NTemplates (see page 605).
- **ADD:** Palmprint support.
- **ADD:** NTemplate can now unpack from NFTemplate, NLTemplate, NETemplate, NFRecord, NLRecord or NERRecord.
- **ADD:** NFTemplate can now unpack from NFRecord.
- **ADD:** NLTemplate can now unpack from NLRecord.
- **ADD:** NETemplate can now unpack from NERRecord.
- **UPD:** Unicode support made cross-platform.

Version 1.4.0.3

- **FIX:** NERRecordClone does not copy Position.

Version 1.4.0.2

- **UPD:** Internal updates.

Version 1.4.0.1

- **FIX:** NTemplateCheck without one or more modality.

Version 1.4.0.0

- **ADD:** NERRecord and NETemplate modules and irises support in NTemplate module in iris-related products.

Version 1.3.0.0

- **UPD:** Merged with NFTemplate, NLTemplate, NFRecord and NLRecord libraries.

Version 1.2.0.1

- **ADD:** NTemplateGetInfo function.

Version 1.2.0.0

- **ADD:** Faces support.
- **UPD:** Updated to use Microsoft Visual C++ Runtime Library 8.0.

Version 1.1.0.0

- **CHN:** Some changes in interface.
- **ADD:** Editing support.

Version 1.0.0.3

- **UPD:** Minor updates.

Version 1.0.0.2

- **FIX:** Minor fixes.

Version 1.0.0.1

- **FIX:** Signature of packed NTemplate.
- **FIX:** NTemplateUnpack function returned error in some cases.
- **FIX:** NTemplateUnpack logic.

Version 1.0.0.0

- Initial release.

6.3.2 .NET API Reference

6.3.2.1 Neurotec Library

Version 3.1.1.0

- **FIX:** Namespaces for some objects.

Version 3.1.0.4

- **FIX:** Marshalling of enumeration arrays in collection classes.

Version 3.1.0.3

- **FIX:** Hide some unused members of NCollection.

Version 3.1.0.2

- **FIX:** Minor fixes.

Version 3.1.0.1

- **UPD:** Updated to reflect changes in unmanaged code.

Version 3.1.0.0

- **ADD:** HXObject operations to NObject (see page 96) class (parameters).
- **ADD:** HNTType, a handle to type information for an object and operations on it.
- **ADD:** New, more specific error codes and exception classes.
- **ADD:** NReadOnlyCollection and NCollection classes and their subclasses as base classes for other NObject (see page 96) internal collections.
- **REM:** Parameter-specific types (now NObject (see page 96) has that functionality).
- **REM:** Obsolete members and types.

Version 3.0.0.1

- **FIX:** Stream wrapper crash sometimes.

Version 3.0.0.0

- **ADD:** System error (errno) intergration for Neurotec (see page 607) exceptions.
- **ADD:** SSSE4, SSE4.1, SSE4.2, SSE4A, LZCNT and POPCNT detection support to NProcessorInfo.
- **UPD:** NLibraryInfo structure.

Version 2.4.5.1

- **FIX:** Minor fixes.

Version 2.4.5.0

- **ADD:** PointD, SizeD, RectangleD types.
- **ADD:** NRange type.

Version 2.4.4.0

- **ADD:** NMemoryStream class.
- **ADD:** NBuffer.Wrap method.

Version 2.4.3.1

- **ADD:** Constants in NResult (see page 118) for defined Neurotec (see page 607) functions return codes.

Version 2.4.2.1

- **ADD:** NBuffer class.
- **ADD:** Extended unmanaged memory management.

Version 2.4.1.1

- **FIX:** Cross-AppDomain usage issues.

Version 2.4.1.0

- **ADD:** NLibraryInfo class.
- **ADD:** NCore.GetInfo method.

Version 2.4.0.0

- **UPD:** Reflects changes in unmanaged code.

Version 2.3.1.0

- **ADD:** NProcessorInfo class.

Version 2.3.0.0

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

Version 2.2.2.0

- **ADD:** More robust unmanaged error handling on Windows.

Version 2.2.1.0

- **ADD:** Classes for internal architecture support.

Version 2.2.0.0

- **UPD:** Updated to support changes in unmanaged code.

Version 2.1.0.0

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

Version 2.0.1.2

- **FIX:** Minor fixes in parameters framework.

Version 2.0.1.1

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

Version 2.0.1.0

- **ADD:** NParameters class for internal infrastructure support.
- **CHN:** Infrastructure optimization for 64-bit support.

Version 2.0.0.0

- **ADD:** A lot of stuff for internal infrastructure support.
- **CHN:** Some changes in internal infrastructure support.

Version 1.0.0.2

- **ADD:** NIndexPair (see page 115) structure.

Version 1.0.0.1

- **FIX:** All error codes are mapped to appropriate exceptions.

Version 1.0.0.0

- Initial release.

6.3.2.2 Neurotec.Biometric.Standards

Version 3.1.3.2

- **UPD:** Minor updates because of Neurotec (see page 607) changes.

Version 3.1.3.1

- **FIX:** Removal from FIRRecord.FingerViews, FMRecord.FingerViews and IIRRecord.IrisImages collections does not remove object from collection.
- **FIX:** Addition to FIRRecord.FingerCollection.FingerViews, FMRecord.FingerCollection.FingerViews and IIRRecord.IrisCollection.IrisImages collections sometimes fails.

Version 3.1.3.0

- **ADD:** Bdif.FlagAllowQuality for use with PIV-profile FIRrecords.
- **FIX:** Removal from FIRRecord.FingerViews, FMRecord.FingerViews and IIRRecord.IrisImages collections.
- **REM:** ANType10Record.MaxSmtDescriptionLength constant.
- **ADD:** ANType9Record.HasMinutiaeRidgeCountsIndicator property.

Version 3.1.2.0

- **CHN:** Rename ANType1Record.*TransmittingResolution constants to more correct names.
- **CHN:** Rename ANRecord.SkipDuplicateFields constant to MergeDuplicateFields and invert its meaning.
- **ADD:** ANType1Record.MaxResolutionV4 constant.

Version 3.1.1.1

- **FIX:** Minor fixes.

Version 3.1.1.0

- **ADD:** ANRecord.FlagRecoverFromBinaryData constant.
- **ADD:** ANTemplate.FlagLeaveInvalidRecordsUnvalidated constant.

Version 3.1.0.0

- **REM:** Obsolete members and types.
- **UPD:** Now uses Neurotec.Images (see page 824) library version 2.6.
- **UPD:** Now uses Neurotec.Biometrics.Templates library version 3.1.

- **UPD:** Now uses Neurotec (see page 607) library version 3.1.

Version 3.0.0.2

- **ADD:** Missing ANSubField and ANField GetValueAsString()/SetValue(string) methods.
- **ADD:** ANRecord.FlagSkipDuplicateFields constant.
- **FIX:** ANSubField and ANField GetValue methods.

Version 3.0.0.1

- **UPD:** Constants from CBEFF register.
- **FIX:** Neurotechnology CBEFF BDB constants.

Version 3.0.0.0

- Initial release of Neurotec.Biometrics.Standards (see page 753) library.
- **ADD:** support for these standards added: ANSI/NIST-ITL 1-2007, ANSI/NIST-ITL 1-2000, ANSI/NIST-ITL 1-1997 and ANSI/NIST-CSL 1-1993.

6.3.2.3 Neurotec.Biometrics.Extractors

Version 3.1.0.0

- **REM:** Obsolete members and types.
- **UPD:** Now uses Neurotec.Images (see page 824) library version 2.6.
- **UPD:** Now uses Neurotec.Biometrics.Templates library version 3.1.
- **UPD:** Now uses Neurotec (see page 607) library version 3.1.

Version 3.0.0.0

- Initial release of Neurotec.Biometrics.Extractors library.
- **CHN:** Previous libraries Neurotec.Biometrics.NEExtractor, Neurotec.Biometrics.NFExtractor (see page 648) and Neurotec.Biometrics.NLExtractor (see page 717) were merged into one Neurotec.Biometrics.Extractors library.
- **ADD:** NFExtractor.MakeSmallTemplate and NFExtractor.IsTemplateSmall methods.
- **ADD:** Palmprint extraction support.

6.3.2.4 Neurotec.Biometrics.Templates

Version 3.1.0.2

- **REM:** NERRecord.GetMaxSize and NLRecord.GetMaxSize dummy methods.

Version 3.1.0.1

- **FIX:** NERRecord constructor signatures.
- **ADD:** CbeffProductType property to NLRecord.

Version 3.1.0.0

- **ADD:** NBiometricType enum.
- **REM:** Obsolete members and types.
- **UPD:** Now uses Neurotec (see page 607) library version 3.1.

Version 3.0.0.1

- **FIX:** NTemplate.RecordCollection.Add() arguments.

Version 3.0.0.0

- **CHN:** Neurotec.Biometrics.NTemplate (see page 726) library renamed to Neurotec.Biometrics.Templates.
- **ADD:** Palmprint support.

6.3.2.5 Neurotec.DeviceManager

Version 3.1.0.1

- **FIX:** Added FPScanners do not trigger events.

Version 3.1.0.0

- **CHN:** IrisCameraEventArgs, IrisCameraEventHandler renamed accordingly to IrisCameraIrisAcquiredEventArgs, IrisCameraIrisAcquiredEventHandler.
- **CHN:** IrisCameraMan.IrisCameras deprecated in favor of new IrisCameraMan.Cameras.
- **REM:** Obsolete members and types.
- **UPD:** Now uses Neurotec.Images (see page 824) library version 2.6.
- **UPD:** Now uses Neurotec (see page 607) library version 3.1.

Version 3.0.1.1

- **FIX:** Camera properties type.

Version 3.0.1.0

- **ADD:** New Camera properties to support IP cameras.

Version 3.0.0.0

- New version of Neurotec.DeviceManager (see page 814) released.
- **MER:** Previous libraries CameraMan, FPScannerMan (see page 143), IrisCameraMan were merged into one library.

6.3.2.6 Neurotec.Images

Version 2.6.0.3

- **FIX:** NImage.GetWrapper and NImage.FromData argument checks.

Version 2.6.0.2

- **UPD:** Minor updates because of Neurotec (see page 607) changes.

Version 2.6.0.1

- **FIX:** GIF open error.

Version 2.6.0.0

- **REM:** Obsolete members and types.
- **UPD:** Now uses Neurotec (see page 607) library version 3.1.

Version 2.5.0.1

- **FIX:** InvalidCastException during GIF file reading.

Version 2.5.0.0

- **ADD:** Lossless JPEG format support.

Version 2.4.1.2

- **FIX:** Minor fixes.

Version 2.4.1.1

- **FIX:** NImageFile disposal errors.

Version 2.4.1.0

- **CHN:** Methods that return System.Byte arrays now return Neurotec.NBuffer.
- **ADD:** NImage methods for image creation from/wrapping of byte array.

Version 2.4.0.0

- **ADD:** PNG format support.
- **ADD:** JPEG2000 format support.

Version 2.3.0.2

- **FIX:** Wrapped images disposing issues.

Version 2.3.0.1

- **FIX:** Cross-AppDomain usage issues.

Version 2.3.0.0

- **ADD:** Image rotation, flipping and cropping functionality.

Version 2.2.1.1

- **ADD:** NImages.GetInfo method.

Version 2.2.1.0

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

Version 2.2.0.0

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

Version 2.1.0.2

- **UPD:** Updated internal structure.

Version 2.1.0.1

- **FIX:** Object disposing issues.

Version 2.1.0.0

- **UPD:** Updated to support changes in unmanaged code.

Version 2.0.3.1

- **FIX:** Minor fixes.

Version 2.0.3.0

- **UPD:** Updated to support changes in unmanaged code.

Version 2.0.2.0

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

Version 2.0.1.0

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

Version 2.0.0.0

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

Version 1.0.0.0

- Initial release.

6.3.2.7 Neurotec.Licensing

Version 3.1.0.0

- **ADD:** NLicense.IsComponentActivated method.
- **UPD:** Now uses Neurotec (🔗 see page 607) library version 3.1.

Version 3.0.0.0

- New version of the Neurotec.Licensing (🔗 see page 895) library released.

6.3.2.8 Neurotec.SmartCards Library

Version 1.2.0.1

- **FIX:** Minor fixes.

Version 1.2.0.0

- **REM:** Obsolete members and types.
- **UPD:** Now uses Neurotec (🔗 see page 607) library version 3.1.

Version 1.1.0.1

- **FIX:** ApduStatus.Is* properties.

Version 1.1.0.0**Version 1.0.0.3**

- **FIX:** BerTlv (🔗 see page 268) and SimpleTlv (🔗 see page 504) SaveMany method signatures.

Version 1.0.0.2

- **FIX:** Cross-AppDomain usage issues.

Version 1.0.0.1

- **FIX:** SCard.TransmitApdu usage of index and responseIndex arguments.

Version 1.0.0.0

- Initial release.

7 Support

Neurotechnology provides customer support during the entire period, while the customer develops and uses his own system using our products. If you face problems using our products, have any questions or suggestions, you can reach us by email: support@neurotechnology.com.

Index

▪

.NET API Reference 1024

.NET Reference 606

A

About 1

AccidentalWhorl enumeration member 749, 806

acsmHeaderAuthenticated enumeration member 381

acsmHeaderNotProcessed enumeration member 381

acsmNone enumeration member 381

acsmProprietary enumeration member 381

Amd enumeration member 638

Amputation enumeration member 749, 806

AnalogueVideo enumeration member 811

Angle enumeration member 749, 814

Ansi enumeration member 813

APDU_INS_ACTIVATE_FILE macro 382

APDU_INS_APPEND_RECORD macro 382

APDU_INS_CHANGE_REFERENCE_DATA macro 382

APDU_INS_CREATE_FILE macro 382

APDU_INS_DEACTIVATE_FILE macro 383

APDU_INS_DELETE_FILE macro 383

APDU_INS_DISABLE_VERIFICATION_REQUIREMENT
macro 383APDU_INS_ENABLE_VERIFICATION_REQUIREMENT
macro 383

APDU_INS_ENVELOPE macro 383

APDU_INS_ENVELOPE_BER_TLV macro 383

APDU_INS_ERASE_BINARY macro 384

APDU_INS_ERASE_BINARY_BER_TLV macro 384

APDU_INS_ERASE_RECORD macro 384

APDU_INS_EXTERNAL_AUTHENTICATE macro 384

APDU_INS_GENERAL_AUTHENTICATE macro 384

APDU_INS_GENERAL_AUTHENTICATE_BER_TLV macro
384APDU_INS_GENERATE_ASYMMETRIC_KEY_PAIR macro
385

APDU_INS_GET_CHALLENGE macro 385

APDU_INS_GET_DATA macro 385

APDU_INS_GET_DATA_BER_TLV macro 385

APDU_INS_GET_RESPONSE macro 385

APDU_INS_INTERNAL_AUTHENTICATE macro 386

APDU_INS_MANAGE_CHANNEL macro 386

APDU_INS_MANAGE_SECURITY_ENVIRONMENT macro
386

APDU_INS_PERFORM_SCQL_OPERATION macro 386

APDU_INS_PERFORM_SECURITY_OPERATION macro 386

APDU_INS_PERFORM_TRANSACTION_OPERATION
macro 386

APDU_INS_PERFORM_USER_OPERATION macro 387

APDU_INS_PUT_DATA macro 387

APDU_INS_PUT_DATA_BER_TLV macro 387

APDU_INS_READ_BINARY macro 387

APDU_INS_READ_BINARY_BER_TLV macro 387

APDU_INS_READ_RECORD macro 387

APDU_INS_READ_RECORD_BER_TLV macro 388

APDU_INS_RESET_RETRY_COUNTER macro 388

APDU_INS_SEARCH_BINARY macro 388

APDU_INS_SEARCH_BINARY_BER_TLV macro 388

APDU_INS_SEARCH_RECORD macro 388

APDU_INS_SELECT macro 389

APDU_INS_TERMINATE_CARD_USAGE macro 389

APDU_INS_TERMINATE_DF macro 389

APDU_INS_TERMINATE_EF macro 389

APDU_INS_UPDATE_BINARY macro 389

APDU_INS_UPDATE_BINARY_BER_TLV macro 389

APDU_INS_UPDATE_RECORD macro 390

APDU_INS_UPDATE_RECORD_BER_TLV macro 390

APDU_INS_VERIFY macro 390

APDU_INS_VERIFY_BER_TLV macro 390

APDU_INS_WRITE_BINARY macro 390

APDU_INS_WRITE_BINARY_BER_TLV macro 390

APDU_INS_WRITE_RECORD macro 391

APDU_STATUS_DATA_STILL_AVAILABLE_XX macro 391

APDU_STATUS_ERROR_CLASS_NOT_SUPPORTED
macro 391APDU_STATUS_ERROR_CNA_AUTHENTICATION_METHO
D_BLOCKED
macro 391APDU_STATUS_ERROR_CNA_COMMAND_INCOMPATIBL
E_WITH_FILE_STRUCTURE
macro 391

APDU_STATUS_ERROR_CNA_COMMAND_NOT_ALLOWE

D	APDU_STATUS_ERROR_SONVMC_XX macro 396
macro 392	APDU_STATUS_ERROR_SONVMU_EXECUTION_ERROR macro 397
APDU_STATUS_ERROR_CNA_CONDITIONS_OF_USE_NO_T_SATISFIED	APDU_STATUS_ERROR_SONVMU_IMMEDIATE_RESPONSE_REQUIRED_BY_THE_CARD
macro 392	macro 397
APDU_STATUS_ERROR_CNA_EXPECTED_SECURE_MESSAGING_DATA_OBJECTS_MISSING	APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_END
macro 392	macro 397
APDU_STATUS_ERROR_CNA_INCORRECT_SECURE_MESSAGING_DATA_OBJECTS	APDU_STATUS_ERROR_SONVMU_TRIGGERING_BY_THE_CARD_START
macro 392	macro 397
APDU_STATUS_ERROR_CNA_NO_INFORMATION	APDU_STATUS_ERROR_SONVMU_XX macro 398
macro 393	APDU_STATUS_ERROR_WP1P2_DF_NAME_ALREADY_EXISTS
APDU_STATUS_ERROR_CNA_REFERENCE_DATA_NOT_USABLE	macro 398
macro 393	APDU_STATUS_ERROR_WP1P2_FILE_ALREADY_EXISTS
APDU_STATUS_ERROR_CNA_SECURITY_STATUS_NOT_SATISFIED	macro 398
macro 393	APDU_STATUS_ERROR_WP1P2_FILE_OR_APPLICATION_NOT_FOUND
APDU_STATUS_ERROR_CNA_XX macro 393	macro 398
APDU_STATUS_ERROR_FICNS_COMMAND_CHAINING_NOT_SUPPORTED	APDU_STATUS_ERROR_WP1P2_FUNCTION_NOT_SUPPORTED
macro 393	macro 398
APDU_STATUS_ERROR_FICNS_LAST_COMMAND_OF_THE_CHAIN_EXPECTED	APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_IN_THE_COMMAND_DATA_FIELD
macro 394	macro 399
APDU_STATUS_ERROR_FICNS_LOGICAL_CHANNEL_NOT_SUPPORTED	APDU_STATUS_ERROR_WP1P2_INCORRECT_PARAMETERS_P1P2
macro 394	macro 399
APDU_STATUS_ERROR_FICNS_NO_INFORMATION	APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_PARAMETERS_P1P2
macro 394	macro 399
APDU_STATUS_ERROR_FICNS_SECURE_MESSAGING_NOT_SUPPORTED	APDU_STATUS_ERROR_WP1P2_NC_INCONSISTENT_WITH_TLV_STRUCTURE
macro 394	macro 399
APDU_STATUS_ERROR_FICNS_XX macro 394	APDU_STATUS_ERROR_WP1P2_NO_INFORMATION
APDU_STATUS_ERROR_INSTRUCTION_NOT_SUPPORTED_OR_INVALID	macro 399
macro 395	APDU_STATUS_ERROR_WP1P2_NO_QUALIFICATION
APDU_STATUS_ERROR_NO_PRECISE_DIAGNOSIS	macro 400
macro 395	APDU_STATUS_ERROR_WP1P2_NOT_ENOUGH_MEMORY_SPACE_IN_THE_FILE
APDU_STATUS_ERROR_PROPRIETARY_67XX macro 395	macro 400
APDU_STATUS_ERROR_PROPRIETARY_6BXX macro 395	APDU_STATUS_ERROR_WP1P2_RECORD_NOT_FOUND
APDU_STATUS_ERROR_PROPRIETARY_6DXX macro 395	macro 400
APDU_STATUS_ERROR_PROPRIETARY_6EXX macro 396	APDU_STATUS_ERROR_WP1P2_REFERENCED_DATA_OR_REFERENCE_DATA_NOT_FOUND
APDU_STATUS_ERROR_PROPRIETARY_6FXX macro 396	macro 400
APDU_STATUS_ERROR_SECURITY_XX macro 396	APDU_STATUS_ERROR_WP1P2_XX macro 400
APDU_STATUS_ERROR_SONVMC_MEMORY_FAILURE	APDU_STATUS_ERROR_WRONG_LE_XX macro 401
macro 396	APDU_STATUS_ERROR_WRONG_LENGTH macro 401
APDU_STATUS_ERROR_SONVMC_NO_INFORMATION	
macro 396	

APDU_STATUS_NO_QUALIFICATION macro 401	IsInterindustryReserved 903
APDU_STATUS_PROPRIETARY_9XXX macro 401	IsLastOrTheOnly 903
APDU_STATUS_WARNING_SONVMC_COUNTER_X macro 401	IsProprietary 903
APDU_STATUS_WARNING_SONVMC_FILE_FILLED_UP_BY_THE_LAST_WRITE macro 402	IsValid 904
APDU_STATUS_WARNING_SONVMC_NO_INFORMATION macro 402	SecureMessaging 904
APDU_STATUS_WARNING_SONVMC_XX macro 402	ToString 901
APDU_STATUS_WARNING_SONVMU_END_OF_FILE_OR_RECORD_BEFORE_READING_NE_BYTES macro 402	ApduClassCreate function 371
APDU_STATUS_WARNING_SONVMU_FILE_CONTROL_INFORMATION_INCORRECTY_FORMATTED macro 402	ApduClassGetChannelNumber function 372
APDU_STATUS_WARNING_SONVMU_NO_INFORMATION macro 403	ApduClassGetSecureMessaging function 372
APDU_STATUS_WARNING_SONVMU_NO_INPUT_DATA_AVAILABLE_FROM_A_SENSOR_ON_THE_CARD macro 403	ApduClassIsInterindustry function 372
APDU_STATUS_WARNING_SONVMU_PART_OF_RETURNED_DATA_MAY_BE_CORRUPTED macro 403	ApduClassIsInterindustryFirst function 373
APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_DEACTIVATED macro 403	ApduClassIsInterindustryFurther function 373
APDU_STATUS_WARNING_SONVMU_SELECTED_FILE_IN_TERMINATION_STATE macro 404	ApduClassIsInterindustryReserved function 373
APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_END macro 404	ApduClassIsLastOrTheOnly function 374
APDU_STATUS_WARNING_SONVMU_TRIGGERING_BY_THE_CARD_START macro 404	ApduClassIsProprietary function 374
APDU_STATUS_WARNING_SONVMU_XX macro 404	ApduClassIsValid function 374
ApduClass structure 899	ApduClassSecureMessaging enumeration 381
!= 901	ApduClassSecureMessaging_ enumeration 381
== 902	ApduInstruction structure 904
about ApduClass structure 899	!= 915
ApduClass 900, 901	== 915
byte 902	about ApduInstruction structure 904
ChannelNumber 902	ActivateFile 908
Equals 901	ApduInstruction 907
GetHashCode 901	AppendRecord 908
IsInterindustry 902	byte 915
IsInterindustryFirst 903	ChangeReferenceData 908
IsInterindustryFurther 903	CreateFile 908
	DeactivateFile 908
	DeleteFile 908
	DisableVerificationRequirement 908
	EnableVerificationRequirement 909
	Envelope 909
	EnvelopeBerTlv 909
	Equals 914
	EraseBinary 909
	EraseBinaryBerTlv 909
	EraseRecord 909
	ExternalAuthenticate 909
	GeneralAuthenticate 909
	GeneralAuthenticateBerTlv 910

GenerateAsymmetricKeyPair 910	ApduStatus structure 916
GetChallenge 910	!= 934
GetData 910	== 934
GetDataBerTlv 910	about ApduStatus structure 916
GetHashCode 914	ApduStatus 924
GetResponse 910	DataStillAvailableValue 935
InternalAuthenticate 910	DataStillAvailableXX 925
IsDataBerTlvEncoded 915	Equals 933
IsValid 916	ErrorClassNotSupported 925
ManageChannel 910	ErrorCnaAuthenticationMethodBlocked 925
ManageSecurityEnvironment 911	ErrorCnaCommandIncompatibleWithFileStructure 925
PerformScqlOperation 911	ErrorCnaCommandNotAllowed 925
PerformSecurityOperation 911	ErrorCnaConditionsOfUseNotSatisfied 925
PerformTransactionOperation 911	ErrorCnaExpectedSecureMessagingDataObjectsMissing 925
PerformUserOperation 911	ErrorCnaIncorrectSecureMessagingDataObjects 925
PutData 911	ErrorCnaNoInformation 926
PutDataBerTlv 911	ErrorCnaReferenceDataNotUsable 926
ReadBinary 911	ErrorCnaSecurityStatusNotSatisfied 926
ReadBinaryBerTlv 911	ErrorCnaXX 926
ReadRecord 912	ErrorFicnsCommandChainingNotSupported 926
ReadRecordBerTlv 912	ErrorFicnsLastCommandOfTheChainExpected 926
ResetRetryCounter 912	ErrorFicnsLogicalChannelNotSupported 926
SearchBinary 912	ErrorFicnsNoInformation 927
SearchBinaryBerTlv 912	ErrorFicnsSecureMessagingNotSupported 927
SearchRecord 912	ErrorFicnsXX 927
Select 912	ErrorInstructionNotSupportedOrInvalid 927
TerminateCardUsage 912	ErrorNoPreciseDiagnosis 927
TerminateDF 913	ErrorProprietary67XX 927
TerminateEF 913	ErrorProprietary6BXX 927
ToString 914	ErrorProprietary6DXX 927
UpdateBinary 913	ErrorProprietary6EXX 928
UpdateBinaryBerTlv 913	ErrorProprietary6FXX 928
UpdateRecord 913	ErrorSecurityXX 928
UpdateRecordBerTlv 913	ErrorSonvmcMemoryFailure 928
Verify 913	ErrorSonvmcNoInformation 928
VerifyBerTlv 913	ErrorSonvmcXX 928
WriteBinary 913	ErrorSonvmuExecutionError 928
WriteBinaryBerTlv 914	ErrorSonvmulImmediateResponseRequiredByTheCard 928
WriteRecord 914	ErrorSonvmuTriggeringByTheCardEnd 929
ApduInstructionIsDataBerTlvEncoded function 375	ErrorSonvmuTriggeringByTheCardStart 929
ApduInstructionIsValid function 375	

ErrorSonvmuTriggeringByTheCardValue 935	IsProcessComplete 939
ErrorSonvmuXX 929	IsProprietary 939
ErrorWP1P2DFNameAlreadyExists 929	IsProprietary9XXX 939
ErrorWP1P2FileAlreadyExists 929	IsStateOfNonVolatileMemoryChanged 939
ErrorWP1P2FileOrApplicationNotFound 929	IsStateOfNonVolatileMemoryUnchanged 939
ErrorWP1P2FunctionNotSupported 929	IsValid 940
ErrorWP1P2IncorrectParametersInTheCommandDataField 929	IsWarningProcessing 940
ErrorWP1P2IncorrectParametersP1P2 930	IsWarningSonvmc 940
ErrorWP1P2NclnconsistentWithParametersP1P2 930	IsWarningSonvmcCounter 940
ErrorWP1P2NclnconsistentWithTlvStructure 930	IsWarningSonvmu 940
ErrorWP1P2NoInformation 930	IsWarningSonvmuTriggeringByTheCard 941
ErrorWP1P2NoQualification 930	NoQualification 931
ErrorWP1P2NotEnoughMemorySpaceInTheFile 930	Proprietary9XXX 931
ErrorWP1P2RecordNotFound 930	SW1 941
ErrorWP1P2ReferencedDataOrReferenceDataNotFound 931	SW2 941
ErrorWP1P2XX 931	ToString 934
ErrorWrongLength 931	ushort 934
ErrorWrongLeValue 935	WarningSonvmcCounterValue 941
ErrorWrongLeXX 931	WarningSonvmcCounterX 931
GetHashCode 933	WarningSonvmcFileFilledUpByTheLastWrite 931
IsCheckingError 935	WarningSonvmcNoInformation 932
IsDataStillAvailable 935	WarningSonvmcXX 932
IsErrorCna 936	WarningSonvmuEndOfFileOrRecordBeforeReadingNeBytes 932
IsErrorFicns 936	WarningSonvmuFileControllInformationIncorrectlyFormatted 932
IsErrorProprietary67XX 936	WarningSonvmuNoInformation 932
IsErrorProprietary6BXX 936	WarningSonvmuNoInputDataAvailableFromASensorOnTheCard 932
IsErrorProprietary6DXX 936	WarningSonvmuPartOfReturnedDataMayBeCorrupted 932
IsErrorProprietary6EXX 937	WarningSonvmuSelectedFileDeactivated 933
IsErrorProprietary6FXX 937	WarningSonvmuSelectedFileInTerminationState 933
IsErrorSecurity 937	WarningSonvmuTriggeringByTheCardEnd 933
IsErrorSonvmc 937	WarningSonvmuTriggeringByTheCardStart 933
IsErrorSonvmu 937	WarningSonvmuTriggeringByTheCardValue 941
IsErrorSonvmuTriggeringByTheCard 937	WarningSonvmuXX 933
IsErrorWP1P2 938	ApduStatusGetDataStillAvailableValue function 375
IsErrorWrongLe 938	ApduStatusGetErrorSonvmuTriggeringByTheCardValue function 376
IsExecutionError 938	ApduStatusGetErrorWrongLeValue function 376
IsInterindustry 938	
IsNormalProcessing 938	
IsProcessAborted 939	

ApduStatusGetSW1 macro 404
 ApduStatusGetSW2 macro 405
 ApduStatusGetWarningSonvmcCounterValue function 376
 ApduStatusGetWarningSonvmuTriggeringByTheCardValue function 377
 ApduStatusIsCheckingError function 377
 ApduStatusIsDataStillAvailable macro 405
 ApduStatusIsErrorCna macro 405
 ApduStatusIsErrorFicns macro 405
 ApduStatusIsErrorProprietary67XX macro 405
 ApduStatusIsErrorProprietary6BXX macro 406
 ApduStatusIsErrorProprietary6DXX macro 406
 ApduStatusIsErrorProprietary6EXX macro 406
 ApduStatusIsErrorProprietary6FXX macro 406
 ApduStatusIsErrorSecurity macro 406
 ApduStatusIsErrorSonvmc macro 407
 ApduStatusIsErrorSonvmu macro 407
 ApduStatusIsErrorSonvmuTriggeringByTheCard macro 407
 ApduStatusIsErrorWP1P2 macro 407
 ApduStatusIsErrorWrongLe macro 407
 ApduStatusIsExecutionError function 377
 ApduStatusIsInterindustry function 378
 ApduStatusIsNormalProcessing function 378
 ApduStatusIsProcessAborted function 378
 ApduStatusIsProcessComplete function 378
 ApduStatusIsProprietary function 379
 ApduStatusIsProprietary9XXX macro 408
 ApduStatusIsStateOfNonVolatileMemoryChanged function 379
 ApduStatusIsStateOfNonVolatileMemoryUnchanged function 379
 ApduStatusIsValid function 380
 ApduStatusIsWarningProcessing function 380
 ApduStatusIsWarningSonvmc macro 408
 ApduStatusIsWarningSonvmcCounter macro 408
 ApduStatusIsWarningSonvmu macro 408
 ApduStatusIsWarningSonvmuTriggeringByTheCard macro 408
 API Reference 13
 Application enumeration member 1001
 Ascending enumeration member 749, 814
 Auburn enumeration member 811

B

Bald enumeration member 811
 Base enumeration member 812
 Bdif class 754
 about Bdif class 754
 FlagDoNotCheckCbeffProductId 755
 Bdif Module 14
 Bdif.h 14
 BDIF_DO_NOT_CHECK_CBEFF_PRODUCT_ID macro 35
 BdifEyeColor enumeration 15
 BdifEyeColor_ enumeration 15
 BdifEyePosition enumeration 17
 BdifEyePosition_ enumeration 17
 BdifFaceExpression enumeration 17
 BdifFaceExpression_ enumeration 17
 BdifFaceFeaturePoint structure 18, 755
 about BdifFaceFeaturePoint structure 755
 BdifFaceFeaturePoint 755
 Code 756
 Y 756
 Type 756
 ToString 756
 X 756
 BdifFaceFeaturePoint_ structure 18
 BdifFaceFeaturePointType enumeration 19
 BdifFaceFeaturePointType_ enumeration 19
 BdifFaceProperties enumeration 19
 BdifFaceProperties_ enumeration 19
 BdifFPatternClass enumeration 21
 BdifFPatternClass_ enumeration 21
 BdifFImpressionType enumeration 22
 BdifFImpressionType_ enumeration 22
 BdifFMinutiaNeighbor structure 25, 756
 about BdifFMinutiaNeighbor structure 756
 BdifFMinutiaNeighbor 757
 Empty 757
 Index 758
 RidgeCount 758
 ToString 757
 BdifFMinutiaNeighbor_ structure 25

BdifFPMinutiaType enumeration 25	BER_TAG_ENUMERATED macro 298
BdifFPMinutiaType_ enumeration 25	BER_TAG_EXTERNAL macro 299
BdifFPPosition enumeration 26	BER_TAG_GENERAL_STRING macro 299
BdifFPPosition_ enumeration 26	BER_TAG_GENERALIZED_TIME macro 299
BdifGender enumeration 28	BER_TAG_GRAPHIC_STRING macro 299
BdifGender_ enumeration 28	BER_TAG_IA5_STRING macro 299
BdifHairColor enumeration 29	BER_TAG_INSTANCE_OF macro 299
BdifHairColor_ enumeration 29	BER_TAG_INTEGER macro 300
BdifImageSourceType enumeration 31	BER_TAG_ISO646_STRING macro 300
BdifImageSourceType_ enumeration 31	BER_TAG_NULL macro 300
BdifIrisOrientation enumeration 32	BER_TAG_NUMERIC_STRING macro 300
BdifIrisOrientation_ enumeration 32	BER_TAG_OBJECT_DESCRIPTOR macro 300
BdifIrisScanType enumeration 33	BER_TAG_OBJECT_IDENTIFIER macro 301
BdifIrisScanType_ enumeration 33	BER_TAG_OCTET_STRING macro 301
BdifScaleUnits enumeration 34	BER_TAG_PRINTABLE_STRING macro 301
BdifScaleUnits_ enumeration 34	BER_TAG_REAL macro 301
BdifStandard enumeration 35	BER_TAG_RELATIVE_OBJECT_IDENTIFIER macro 301
BdifStandard_ enumeration 35	BER_TAG_SEQUENCE macro 301
BdifTypes Module 14	BER_TAG_SEQUENCE_OF macro 302
BdifTypes.h 36	BER_TAG_SET macro 302
Beard enumeration member 805	BER_TAG_SET_OF macro 302
becBlack enumeration member 15	BER_TAG_T61_STRING macro 302
becBlue enumeration member 15	BER_TAG_TELETEX_STRING macro 302
becBrown enumeration member 15	BER_TAG_UNIVERSAL_STRING macro 302
becGray enumeration member 15	BER_TAG_UTC_TIME macro 303
becGreen enumeration member 15	BER_TAG_UTF8_STRING macro 303
becHazel enumeration member 15	BER_TAG_VIDEOTEX_STRING macro 303
becMaroon enumeration member 15	BER_TAG_VISIBLE_STRING macro 303
becMultiColored enumeration member 15	BerTag structure 941
becPink enumeration member 15	!= 951
becUnknown enumeration member 15	== 951
becUnspecified enumeration member 15	about BerTag structure 941
bepLeft enumeration member 17	BerTag 944
bepRight enumeration member 17	BmpString 945
bepUnknown enumeration member 17	Boolean 945
BER_TAG_BIT_STRING macro 297	CharacterString 945
BER_TAG_BMP_STRING macro 298	Class 952
BER_TAG_BOOLEAN macro 298	EmbeddedPdv 945
BER_TAG_CHARACTER_STRING macro 298	Encoding 952
BER_TAG_EMBEDDED_PDV macro 298	Enumerated 945
BER_TAG_END_OF_CONTENTS macro 298	Equals 948

- External 945
- GeneralizedTime 945
- GeneralString 945
- GetHashCode 949
- GraphicString 946
- IA5String 946
- InstanceOf 946
- int 951
- Integer 946
- Iso646String 946
- IsValid 949
- Length 952
- Null 946
- Number 952
- NumericString 946
- ObjectDescriptor 946
- ObjectIdentifier 946
- OctetString 947
- PrintableString 947
- Read 949, 950
- Real 947
- RelativeObjectIdentifier 947
- Save 950
- Sequence 947
- SequenceOf 947
- Set 947
- SetOf 947
- T61String 947
- TeletexString 948
- ToString 950
- UniversalString 948
- UtcTime 948
- Utf8String 948
- VideotexString 948
- VisibleString 948
- Write 950
- BerTagClass enumeration 297
- BerTagClass_ enumeration 297
- BerTagCreate function 272
- BerTagEncoding enumeration 297
- BerTagEncoding_ enumeration 297
- BerTagGetClass function 273
- BerTagGetEncoding function 273
- BerTagGetLength function 273
- BerTagGetNumber function 274
- BerTagIsValid function 274
- BerTagRead function 274
- BerTagWrite function 275
- BerTlv class 952
 - about BerTlv class 952
 - Clone 953
 - Create 954
 - FromHandle 955
 - FromMemory 955, 956, 957
 - GetSizeMany 957
 - Length 961
 - LoadMany 957, 958, 959
 - MinLengthSize 961
 - Save 959, 960
 - SaveMany 960
 - Size 961
 - Tag 961
- BerTlv Module 268
- BerTlv.h 303
- BerTlvClone function 275
- BerTlvCreate function 276
- BerTlvCreateFromMemory function 276
- BerTlvGetLength function 277
- BerTlvGetMinLengthSize function 277
- BerTlvGetSize function 278
- BerTlvGetSizeMany function 278
- BerTlvGetTag function 279
- BerTlvLoadManyFromMemory function 279
- BerTlvSaveManyToMemory function 280
- BerTlvSaveToMemory function 280
- BerTlvSetMinLengthSize function 281
- bfeEyesAway enumeration member 17
- bfeFrowning enumeration member 17
- bfeNeutral enumeration member 17
- bfeRaisedBrows enumeration member 17
- bfeSmile enumeration member 17
- bfeSmileOpenedJaw enumeration member 17

bfeSquinting enumeration member 17	bfpitLiveScanOpticalContactlessRolled enumeration member 22
bfeUnspecified enumeration member 17	bfpitLiveScanOpticalContactPlain enumeration member 22
bffptPoint2D enumeration member 19	bfpitLiveScanOpticalContactRolled enumeration member 22
bfpBeard enumeration member 19	bfpitLiveScanPalm enumeration member 22
bfpBlink enumeration member 19	bfpitLiveScanPlain enumeration member 22
bfpBothEyePatch enumeration member 19	bfpitLiveScanRolled enumeration member 22
bfpcAccidentalWhorl enumeration member 21	bfpitLiveScanVerticalSwipe enumeration member 22
bfpcAmputation enumeration member 21	bfpitNonliveScanPalm enumeration member 22
bfpcCentralPocketLoop enumeration member 21	bfpitNonliveScanPlain enumeration member 22
bfpcDoubleLoop enumeration member 21	bfpitNonliveScanRolled enumeration member 22
bfpcLeftSlantLoop enumeration member 21	bfpitOther enumeration member 22
bfpcPlainArch enumeration member 21	bfpitUnknown enumeration member 22
bfpcPlainWhorl enumeration member 21	bfpLeftEyePatch enumeration member 19
bfpcRadialLoop enumeration member 21	bfpMoustache enumeration member 19
bfpcRightSlantLoop enumeration member 21	bfpMouthOpen enumeration member 19
bfpcScar enumeration member 21	bfpmtBifurcation enumeration member 25
bfpcTentedArch enumeration member 21	bfpmtEnd enumeration member 25
bfpcUlnarLoop enumeration member 21	bfpmtOther enumeration member 25
bfpcUnknown enumeration member 21	bfpmtUnknown enumeration member 25
bfpcWhorl enumeration member 21	bfpmtUnspecified enumeration member 25
bfpDarkGlasses enumeration member 19	bfpNoEar enumeration member 19
bfpDistortingCondition enumeration member 19	bfpNotSpecified enumeration member 19
bfpGlasses enumeration member 19	bfppLeftFullPalm enumeration member 26
bfpHat enumeration member 19	bfppLeftHypothenar enumeration member 26
bfpitLatentImpression enumeration member 22	bfppLeftIndexFinger enumeration member 26
bfpitLatentLift enumeration member 22	bfppLeftInterdigital enumeration member 26
bfpitLatentPalmImpression enumeration member 22	bfppLeftLittleFinger enumeration member 26
bfpitLatentPalmLift enumeration member 22	bfppLeftLowerPalm enumeration member 26
bfpitLatentPalmPhoto enumeration member 22	bfppLeftMiddleFinger enumeration member 26
bfpitLatentPalmTracing enumeration member 22	bfppLeftOther enumeration member 26
bfpitLatentPhoto enumeration member 22	bfppLeftRingFinger enumeration member 26
bfpitLatentTracing enumeration member 22	bfppLeftThenar enumeration member 26
bfpitLiveScanContactless enumeration member 22	bfppLeftThumb enumeration member 26
bfpitLiveScanNonOpticalContactlessPlain enumeration member 22	bfppLeftUpperPalm enumeration member 26
bfpitLiveScanNonOpticalContactlessRolled enumeration member 22	bfppLeftWritersPalm enumeration member 26
bfpitLiveScanNonOpticalContactPlain enumeration member 22	bfppMajorCase enumeration member 26
bfpitLiveScanNonOpticalContactRolled enumeration member 22	bfppPlainLeftFourFingers enumeration member 26
bfpitLiveScanOpticalContactlessPlain enumeration member 22	bfppPlainLeftThumb enumeration member 26
	bfppPlainRightFourFingers enumeration member 26
	bfppPlainRightThumb enumeration member 26
	bfppPlainThumbs enumeration member 26

bfppRightFullPalm enumeration member 26	BigEndian enumeration member 895
bfppRightHypothenar enumeration member 26	Binarized enumeration member 747
bfppRightIndexFinger enumeration member 26	bioBase enumeration member 32
bfppRightInterdigital enumeration member 26	bioFlipped enumeration member 32
bfppRightLittleFinger enumeration member 26	bioUndefined enumeration member 32
bfppRightLowerPalm enumeration member 26	bistAnalogueVideo enumeration member 31
bfppRightMiddleFinger enumeration member 26	bistCorrected enumeration member 33
bfppRightOther enumeration member 26	bistDigitalPhoto enumeration member 31
bfppRightRingFinger enumeration member 26	bistDigitalVideo enumeration member 31
bfppRightThenar enumeration member 26	bistInterlaceField enumeration member 33
bfppRightThumb enumeration member 26	bistInterlaceFrame enumeration member 33
bfppRightUpperPalm enumeration member 26	bistProgressive enumeration member 33
bfppRightWritersPalm enumeration member 26	bistScannedPhoto enumeration member 31
bfppUnknown enumeration member 26	bistUndefined enumeration member 33
bfppUnknownPalm enumeration member 26	bistUnknown enumeration member 31
bfpRightEyePatch enumeration member 19	bistUnknownPhoto enumeration member 31
bfpScarf enumeration member 19	bistUnknownVideo enumeration member 31
bfpSpecified enumeration member 19	bistUnspecified enumeration member 31
bfpTeethVisible enumeration member 19	bistVendor enumeration member 31
bgFemale enumeration member 28	Black enumeration member 804, 811
bgMale enumeration member 28	Blink enumeration member 805
bgUnknown enumeration member 28	Blonde enumeration member 811
bgUnspecified enumeration member 28	Blue enumeration member 804, 811
bhcAuburn enumeration member 29	Bmp class 825
bhcBald enumeration member 29	about Bmp class 825
bhcBlack enumeration member 29	LoadImage 825, 826
bhcBlonde enumeration member 29	LoadImageFromBitmap 826
bhcBlue enumeration member 29	LoadImageFromHBitmap 827
bhcBrown enumeration member 29	SaveImage 827, 828
bhcGray enumeration member 29	SaveImageToBitmap 828
bhcGreen enumeration member 29	SaveImageToHBitmap 828
bhcOrange enumeration member 29	Bmp Module 189
bhcPink enumeration member 29	Bmp.h 194
bhcPurple enumeration member 29	BmpLoadImageFromFile function 189
bhcRed enumeration member 29	BmpLoadImageFromHBitmap function 190
bhcSandy enumeration member 29	BmpLoadImageFromMemory function 190
bhcStrawberry enumeration member 29	BmpLoadImageFromStream function 191
bhcUnknown enumeration member 29	BmpSaveImageToFile function 191
bhcUnspecified enumeration member 29	BmpSaveImageToHBitmap function 192
bhcWhite enumeration member 29	BmpSaveImageToMemory function 192
Bifurcation enumeration member 749, 808	BmpSaveImageToStream function 193

BodyOdor enumeration member 1009
 BothEyePatch enumeration member 805
 BottomUp enumeration member 884
 Brown enumeration member 804, 811
 bsAnsi enumeration member 35
 bsIso enumeration member 35
 bsuNone enumeration member 34
 bsuPixelsPerCentimeter enumeration member 34
 bsuPixelsPerInch enumeration member 34

C

C/C++ API Reference 1011
 C/C++ Reference 13
 CartesianYX enumeration member 749, 814
 CartesianXY enumeration member 749, 814
 CBEFF_BDBFI_ID3_SEMICONDUCTORS_COMPACT_CARD_EXTENSION
 macro 41
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FACE_IMAGE
 macro 41
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_IMAGE
 macro 41
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_MINUTIAE_CARD_COMPACT_N
 macro 41
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_MINUTIAE_CARD_COMPACT_V
 macro 41
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_MINUTIAE_CARD_NORMAL_N
 macro 42
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_MINUTIAE_CARD_NORMAL_V
 macro 42
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_MINUTIAE_RECORD_N
 macro 42
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_MINUTIAE_RECORD_X
 macro 42
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_PATTERN_SPECTRAL_DISCRETE_FOURIER_TRANSFORM
 macro 43
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_PATTERN_SPECTRAL_GABOR_FILTER
 macro 43
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_FIN
 GER_PATTERN_SPECTRAL_QUANTIZED_CO_SINUSOIDAL_TRIPLET
 macro 43
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS
 _IMAGE_POLAR
 macro 43
 CBEFF_BDBFI_ISO_IEC_JTC_1_SC_37_BIOMETRICS_IRIS
 _IMAGE_RECTILINEAR
 macro 44
 CBEFF_BDBFI_NEUROTECHNOLOGIJA_NLRECORD_3
 macro 44
 CBEFF_BDBFI_NEUROTECHNOLOGIJA_NTEMPLATE
 macro 44
 CBEFF_BO_123ID macro 51
 CBEFF_BO_A_4_VISION macro 51
 CBEFF_BO_ACSYS_BIOMETRICS macro 51
 CBEFF_BO_ACTIVCARD macro 51
 CBEFF_BO_ANTHEUS_TECHNOLOGY macro 51
 CBEFF_BO_ATMEL macro 51
 CBEFF_BO_AURORA_WIRELESS_TECHNOLOGIES macro 52
 CBEFF_BO_AUTHEN_TEC macro 52
 CBEFF_BO_AWARE macro 52
 CBEFF_BO_BIO_WISE macro 52
 CBEFF_BO_BIOKEY_INTERNATIONAL macro 52
 CBEFF_BO_BIOKEY_INTERNATIONAL_2 macro 52
 CBEFF_BO_BIOLINK_TECHNOLOGIES_INTERNATIONAL
 macro 53
 CBEFF_BO_BIOSCRYPT macro 53
 CBEFF_BO_BIOVISION macro 53
 CBEFF_BO_CYBER_SIGN macro 53
 CBEFF_BO_COGENT_SYSTEMS macro 53
 CBEFF_BO_CRYPTOMETRICS macro 54
 CBEFF_BO_CROSS_MATCH_TECHNOLOGIES macro 54
 CBEFF_BO_DERMALOG_IDENTIFICATION_SYSTEMS
 macro 54
 CBEFF_BO_DIGITALPERSONA macro 54
 CBEFF_BO_EAST_SHORE_TECHNOLOGIES macro 54
 CBEFF_BO_ECRYPT macro 54
 CBEFF_BO_FINGERPRINT_CARDS macro 55
 CBEFF_BO_GERMAN_INSTITUTE_FOR_STANDARDIZATION
 macro 55
 CBEFF_BO_GREEN_BIT_AMERICAS macro 55
 CBEFF_BO_GRIAULE_TECNOLOGIA macro 55

CBEFF_BO_GUARDWARE_SYSTEMS macro 55	CBEFF_BO_VALIDITY macro 62
CBEFF_BO_HITACHI macro 55	CBEFF_BO_VERIDICOM macro 63
CBEFF_BO_HUMANSKAN macro 56	CBEFF_BO_VERIDT macro 63
CBEFF_BO_ID3_SEMICONDUCTORS macro 56	CBEFF_BO_VIISAGE macro 63
CBEFF_BO_IDENTIX macro 56	CBEFF_BO_XTEC macro 63
CBEFF_BO_IDENTIX_2 macro 56	CbeffBdbFormatIdentifiers class 758
CBEFF_BO_IMAGE_WARE_SYSTEMS macro 56	about CbeffBdbFormatIdentifiers class 758
CBEFF_BO_INCITS_TC_M1_BIOMETRICS macro 57	BioLogicaSistemasIso19794_2BioUniStation 766
CBEFF_BO_INFINEON_TECHNOLOGIES macro 57	BioLogicaSistemasIso19794_4BioUniStation 766
CBEFF_BO_INNOVATRICS macro 57	BioLogicaSistemasIso19794_5BioUniStation 766
CBEFF_BO_INTERNATIONAL_BIOMETRIC_GROUP macro 57	FederalOfficeForInformationSecurityTRBiometricsXmlFac e10 766
CBEFF_BO_INVESTORSHELPERS macro 57	FederalOfficeForInformationSecurityTRBiometricsXmlFing er10 766
CBEFF_BO_IRIDIAN_TECHNOLOGIES macro 57	Id3SemiconductorsCompactCardExtension 767
CBEFF_BO_ISO_IEC_JTC_1_SC_37_BIOMETRICS macro 58	IncitsTCM1BiometricsAviMedia 767
CBEFF_BO_JANUS_ASSOCIATES macro 58	IncitsTCM1BiometricsBiometricFusionData 767
CBEFF_BO_LG_ELECTRONICS_USA macro 58	IncitsTCM1BiometricsBmplImage 767
CBEFF_BO_LOGICO_SMARTCARD_SOLUTIONS macro 58	IncitsTCM1BiometricsEbts 767
CBEFF_BO_LUMIDIGM macro 58	IncitsTCM1BiometricsFacelImage 767
CBEFF_BO_NEC_SOLUTIONS_AMERICA macro 58	IncitsTCM1BiometricsFingerImage 767
CBEFF_BO_NEUROTECHNOLOGIJA macro 59	IncitsTCM1BiometricsFingerMinutiaeN 767
CBEFF_BO_NIST macro 59	IncitsTCM1BiometricsFingerMinutiaeU 768
CBEFF_BO_NITGEN macro 59	IncitsTCM1BiometricsFingerMinutiaeX 768
CBEFF_BO_NOT_FOR_USE macro 59	IncitsTCM1BiometricsFingerPatternN 768
CBEFF_BO_OMNIPERCEPTION macro 59	IncitsTCM1BiometricsGenericFbiEftsRecord 768
CBEFF_BO_PRECISE_BIOMETRICS macro 60	IncitsTCM1BiometricsGif89aImage 768
CBEFF_BO_PRIVATE macro 60	IncitsTCM1BiometricsHandGeometryN 768
CBEFF_BO_RECOGNITION_SYSTEMS macro 60	IncitsTCM1BiometricsHandGeometryX 768
CBEFF_BO_SAFLINK macro 60	IncitsTCM1BiometricsIrisPolar 768
CBEFF_BO_SAGEM_MORPHO macro 60	IncitsTCM1BiometricsIrisRectilinear 769
CBEFF_BO_SECUGEN macro 60	IncitsTCM1BiometricsJpeg2000Image 769
CBEFF_BO_SECURE_DESIGN macro 61	IncitsTCM1BiometricsJpegImage 769
CBEFF_BO_SILEX_TECHNOLOGY macro 61	IncitsTCM1BiometricsMpeg1Part3Media 769
CBEFF_BO_SONDA_TECHNOLOGIES macro 61	IncitsTCM1BiometricsMpegMedia 769
CBEFF_BO_STARTEK_ENGINEERING macro 61	IncitsTCM1BiometricsNiem 769
CBEFF_BO_SUPREMA macro 61	IncitsTCM1BiometricsNistItl1_2000Type10Record 769
CBEFF_BO_TESTING_1 macro 61	IncitsTCM1BiometricsNistItl1_2000Type13Record 769
CBEFF_BO_TESTING_2 macro 62	IncitsTCM1BiometricsNistItl1_2000Type14Record 770
CBEFF_BO_THALES_IDENTIFICATION macro 62	IncitsTCM1BiometricsNistItl1_2000Type15Record 770
CBEFF_BO_ULTRA_SCAN macro 62	IncitsTCM1BiometricsNistItl1_2000Type16Record 770
CBEFF_BO_UPEK macro 62	

IncitsTCM1BiometricsNistItl1_2000Type4Record 770	773
IncitsTCM1BiometricsNistItl1_2007Type10Record 770	IsolecJtc1SC37BiometricsFingerMinutiaeRecordN 774
IncitsTCM1BiometricsNistItl1_2007Type14Record 770	IsolecJtc1SC37BiometricsFingerMinutiaeRecordX 774
IncitsTCM1BiometricsNistItl1_2007Type17Record 770	IsolecJtc1SC37BiometricsFingerPatternSkeletalDataCard Compact 774
IncitsTCM1BiometricsNistItl1_200XRecordCollectionForTenPrintCapture 771	IsolecJtc1SC37BiometricsFingerPatternSkeletalDataCard Normal 774
IncitsTCM1BiometricsPngImage 771	IsolecJtc1SC37BiometricsFingerPatternSkeletalDataRecord 774
IncitsTCM1BiometricsSignatureSignCommonFeatureDataN 771	IsolecJtc1SC37BiometricsFingerPatternSpectralDiscreteFourierTransform 774
IncitsTCM1BiometricsSignatureSignCommonFeatureDataX 771	IsolecJtc1SC37BiometricsFingerPatternSpectralGaborFilter 774
IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataN 771	IsolecJtc1SC37BiometricsFingerPatternSpectralQuantizedCOSinusoidalTriplet 775
IncitsTCM1BiometricsSignatureSignRawAndCommonFeatureDataX 771	IsolecJtc1SC37BiometricsHandGeometrySilhouette 775
IncitsTCM1BiometricsSignatureSignRawDataN 771	IsolecJtc1SC37BiometricsIrisImagePolar 775
IncitsTCM1BiometricsSignatureSignRawDataX 772	IsolecJtc1SC37BiometricsIrisImageRectilinear 775
IncitsTCM1BiometricsTiffImage 772	IsolecJtc1SC37BiometricsSignatureSignTimeSeriesCompact 775
IncitsTCM1BiometricsVrml3DObjectData 772	IsolecJtc1SC37BiometricsSignatureSignTimeSeriesFull 775
IncitsTCM1BiometricsWavAudio 772	IsolecJtc1SC37BiometricsVascularImageData 775
IncitsTCM1BiometricsWsqImage 772	NeurotechnologijaNLRecord3 776
IsolecJtc1SC27ITSecurityTechniquesBiometricReferenceTemplateCertificate 772	NeurotechnologijaNTemplate 776
IsolecJtc1SC37BiometricsFacelImage 772	CbeffBdbFormatIdentifiers Module 37
IsolecJtc1SC37BiometricsFingerImage 772	CbeffBdbFormatIdentifiers.h 44
IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactN 773	CbeffBiometricOrganizations class 776
IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactRH 773	_123ID 779
IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactV 773	A4Vision 779
IsolecJtc1SC37BiometricsFingerMinutiaeCardCompactVH 773	about CbeffBiometricOrganizations class 776
IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalN 773	AcSysBiometrics 779
IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalRNH 773	ActivCard 779
IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalV 773	AntheusTechnology 779
IsolecJtc1SC37BiometricsFingerMinutiaeCardNormalVNH 773	AOptix 779
	Atmel 779
	AuroraWirelessTechnologies 780
	AuthenTec 780
	Aware 780

BioKeyInternational 780	Lumidigm 784
BioKeyInternational2 780	NecSolutionsAmerica 785
BioLinkTechnologiesInternational 780	Neurotechnologija 785
BioLogicaSistemas 780	Nist 785
Bioscrypt 780	Nitgen 785
Biovision 780	NotForUse 785
BioWise 781	OmniMotorola 785
CyberSign 781	OmniPerception 785
CogentSystems 781	PreciseBiometrics 785
CryptoMetrics 781	Private 785
CrossMatchTechnologies 781	RecognitionSystems 786
DermalogIdentificationSystems 781	Saflink 786
Digent 781	SagemMorpho 786
DigitalPersona 781	SecuGen 786
EasternGoldenFingerTechnologyBeijing 781	SecureDesign 786
EastShoreTechnologies 782	SilexTechnology 786
ECryp 782	SondaTechnologies 786
FederalOfficeForInformationSecurity 782	StarTekEngineering 786
FingerprintCards 782	Suprema 786
Fujitsu 782	Testing1 787
GermanInstituteForStandardization 782	Testing2 787
GreenBitAmericas 782	ThalesIdenitfication 787
GriauleTecnologia 782	UltraScan 787
GuardwareSystems 782	Upek 787
Hitachi 783	Validity 787
HumanScan 783	Veridicom 787
Id3Semiconductors 783	Veridt 787
Identix 783	Viisage 787
Identix2 783	WarwickWarp 788
ImageWareSystems 783	XTec 788
IncitsTCM1Biometrics 783	CbeffBiometricOrganizations Module 48
InfineonTechnologies 783	CbeffBiometricOrganizations.h 63
Innovatrics 783	CellQuality enumeration member 813
InternationalBiometricGroup 784	Centaur enumeration member 638
InvestorsHelpers 784	CentralPocketLoop enumeration member 749, 806
IridianTechnologies 784	Change Log 1010
IsolecJtc1SC27ITSecurityTechniques 784	Cyrix enumeration member 638
IsolecJtc1SC37Biometrics 784	Constructed enumeration member 1002
JanusAssociates 784	ConstructedBerTlv class 961
LGElectronicsUsa 784	about ConstructedBerTlv class 961
LogicoSmartcardSolutions 784	DataObjects 966

- ConstructedBerTlv.DataObjectCollection class 963
 - about ConstructedBerTlv.DataObjectCollection class 963
 - Add 963, 964, 965
 - Contains 966
 - IndexOf 966
- ConstructedBerTlvAddDataObject function 281
- ConstructedBerTlvAddDataObjectCopy function 282
- ConstructedBerTlvAddDataObjectFromMemory function 282
- ConstructedBerTlvClearDataObjects function 283
- ConstructedBerTlvFindDataObject function 283
- ConstructedBerTlvFindDataObjectIndex function 284
- ConstructedBerTlvGetDataObject function 285
- ConstructedBerTlvGetDataObjectCapacity function 285
- ConstructedBerTlvGetDataObjectCount function 286
- ConstructedBerTlvRemoveDataObject function 286
- ConstructedBerTlvSetDataObjectCapacity function 287
- ContextSpecific enumeration member 1001
- CorePoints enumeration member 813
- Corrected enumeration member 812
- D**
- DarkGlasses enumeration member 805
- DeltaPoints enumeration member 813
- Descending enumeration member 749, 814
- DigitalPhoto enumeration member 811
- DigitalVideo enumeration member 811
- DistortingCondition enumeration member 805
- Dna enumeration member 1009
- DoubleLoop enumeration member 749, 806
- E**
- EarShape enumeration member 1009
- EightNeighbors enumeration member 752
- EightNeighborsWithIndexes enumeration member 752
- EyesAway enumeration member 805
- EyesNotDetected enumeration member 752
- End enumeration member 749, 808
- F**
- FaceNotDetected enumeration member 752
- FaceTooCloseToImageBorder enumeration member 752
- FacialFeatures enumeration member 1009
- Female enumeration member 810
- FingerGeometry enumeration member 1009
- Fingerprint enumeration member 1009
- FlipY enumeration member 885
- FlipNone enumeration member 885
- Flipped enumeration member 812
- FlipX enumeration member 885
- FlipXY enumeration member 885
- FMC_BDT_TAG_CELL_QUALITY_DATA macro 70
- FMC_BDT_TAG_CORE_POINT_DATA macro 70
- FMC_BDT_TAG_DELTA_POINT_DATA macro 71
- FMC_BDT_TAG_FINGER_MINUTIAE_DATA macro 71
- FMC_BDT_TAG_RIDGE_COUNT_DATA macro 71
- FMC_BIT_BHT_BAPT_TAG_FEATURE_HANDLING_INDICATOR macro 71
- FMC_BIT_BHT_BAPT_TAG_MINUTIAE_ORDER macro 71
- FMC_BIT_BHT_BAPT_TAG_NUMBER_OF_MINUTIAE macro 71
- FMC_DEFAULT_MAX_ENROLL_MC macro 72
- FMC_DEFAULT_MAX_VERIFY_MC macro 72
- FMC_DEFAULT_MIN_ENROLL_MC macro 72
- FMC_DEFAULT_MIN_VERIFY_MC macro 72
- FMCard class 788
 - about FMCard class 788
 - DefaultMaxEnrollMC 788
 - DefaultMaxVerifyMC 789
 - DefaultMinEnrollMC 789
 - DefaultMinVerifyMC 789
 - ReadCompactSizeMinutiae 789, 790, 791
 - ReadNormalSizeMinutiae 791, 792, 793
 - WriteCompactSizeMinutiae 793, 794, 795, 796, 797
 - WriteNormalSizeMinutiae 797, 798, 799, 800
- FMCard Module 66
- FMCard.h 72
- FMCardWriteCompactSizeMinutiae function 67
- FMCardWriteNormalSizeMinutiae function 68
- FmcBdtTags class 801
 - about FmcBdtTags class 801
 - CellQualityData 802

- CorePointData 802
- DeltaPointData 802
- FingerMinutiaeData 802
- RidgeCountData 802
- FmcBitBhtBaptTags class 802
 - about FmcBitBhtBaptTags class 802
- FeatureHandlingIndicator 803
- MinutiaeOrder 803
- NumberOfMinutiae 803
- FmcFeatureHandling 69
- FmcFeatureHandling enumeration 69
- fmcfhCellQuality enumeration member 69
- fmcfhCorePoints enumeration member 69
- fmcfhDeltaPoints enumeration member 69
- fmcfhNone enumeration member 69
- fmcfhRidgeCounts enumeration member 69
- FmcMinutiaOrder 69
- FmcMinutiaOrder enumeration 69
- fmcmoAngle enumeration member 69
- fmcmoAscending enumeration member 69
- fmcmoCartesianYX enumeration member 69
- fmcmoCartesianXY enumeration member 69
- fmcmoDescending enumeration member 69
- fmcmoNone enumeration member 69
- fmcmoPolar enumeration member 69
- fmcmoXOrYCoordinateExtension enumeration member 69
- FootPrint enumeration member 1009
- FourNeighbors enumeration member 752
- FourNeighborsWithIndexes enumeration member 752
- FPScanner class 815
 - about FPScanner class 815
 - FingerPlaced 817
 - FingerRemoved 817
 - Id 817
 - ImageScanned 818
 - IsCapturing 817
 - IsCapturingChanged 818
 - Owner 817
 - StartCapturing 816
 - StartCapturingForOneImage 816
 - StopCapturing 816
 - ToString 817
- FPScanner Module 137
- FPScanner.h 142
- FPScannerGetIdA function 137
- FPScannerGetIdW function 138
- FPScannerImageScannedEventArgs class 818
 - about FPScannerImageScannedEventArgs class 818
- FPScannerImageScannedEventArgs 818
- Image 818
- FPScannerIsCapturing function 138
- FPScannerMan class 819
 - about FPScannerMan class 819
- FPScannerMan 820
- GetAvailableModules 821
- ScannerAdded 821
- ScannerRemoved 822
- Scanners 821
- FPScannerMan Module 143
- FPScannerMan.FPScannerCollection class 820
 - about FPScannerMan.FPScannerCollection class 820
- IndexOf 820
- this 821
- FPScannerMan.h 150
- FPScannerManGetAvailableModulesA function 144
- FPScannerManGetAvailableModulesW function 144
- FPScannerManGetScanner function 144
- FPScannerManGetScannerByIdA function 145
- FPScannerManGetScannerByIdW function 145
- FPScannerManGetScannerCount function 146
- FPScannerManInitialize function 146
- FPScannerManInitializeWithModulesA function 147
- FPScannerManInitializeWithModulesW function 147
- FPScannerManScannerEventArgs class 822
 - about FPScannerManScannerEventArgs class 822
- FPScannerManScannerEventArgs 822
- Scanner 822
- FPScannerManSetScannerAddedCallbackA function 148
- FPScannerManSetScannerAddedCallbackW function 148
- FPScannerManSetScannerRemovedCallbackA function 149
- FPScannerManSetScannerRemovedCallbackW function 149
- FPScannerManUninitialize function 149

FPScannerSetFingerPlacedCallback function 139
 FPScannerSetFingerRemovedCallback function 139
 FPScannerSetImageScannedCallback function 140
 FPScannerSetIsCapturingChangedCallback function 140
 FPScannerStartCapturing function 140
 FPScannerStartCapturingForOneImage function 141
 FPScannerStopCapturing function 141
 Frowning enumeration member 805

G

Gait enumeration member 1009
 GeneralizationFailed enumeration member 752
 Glasses enumeration member 805
 Gray enumeration member 804, 811
 Green enumeration member 804, 811

H

HandGeometry enumeration member 1009
 HasCurvature enumeration member 748
 HasG enumeration member 748
 HasQuality enumeration member 748
 Hat enumeration member 805
 Hazel enumeration member 804
 HBerTlv 297
 HeaderAuthenticated enumeration member 1001
 HeaderNotProcessed enumeration member 1001
 hImage variable 142
 HNFExtractor 161
 HNFRecord 571
 HNFTemplate 604
 HNIImage 221
 HNIImageFile 226
 HNIImageFormat 259
 HNLExtractor 180
 hScanner variable 150
 HSimpleTlv 513

I

IHead Module 194
 IHead.h 198

IHeadLoadImageFromFile function 195
 IHeadLoadImageFromMemory function 195
 IHeadLoadImageFromStream function 196
 IHeadSaveImageToFile function 196
 IHeadSaveImageToMemory function 197
 IHeadSaveImageToStream function 197
 INeurotecException interface 637
 about INeurotecException interface 637
 Code 638
 ManagedStackTrace 638
 UnmanagedStackTrace 638
 Intel enumeration member 638
 InterlaceField enumeration member 812
 InterlaceFrame enumeration member 812
 Iris enumeration member 1009
 Iso enumeration member 813

J

Java Card Part 6
 JC Bio API Matching On-Card Library Configurations 8
 Jpeg class 828
 about Jpeg class 828
 DefaultQuality 829
 LoadImage 829, 830
 SaveImage 830, 831
 Jpeg Module 198
 Jpeg.h 204
 JPEG_DEFAULT_QUALITY macro 204
 JpegLoadImageFromFile function 199
 JpegLoadImageFromMemory function 199
 JpegLoadImageFromStream function 200
 JpegSaveImageToFile function 201
 JpegSaveImageToMemory function 201
 JpegSaveImageToStream function 202

K

KeystrokeDynamics enumeration member 1009

L

Large enumeration member 747, 753

LatentImpression enumeration member 747, 807
 LatentLift enumeration member 747, 807
 LatentPalmImpression enumeration member 747, 807
 LatentPalmLift enumeration member 747, 807
 LatentPalmPhoto enumeration member 747, 807
 LatentPalmTracing enumeration member 747, 807
 LatentPhoto enumeration member 747, 807
 LatentTracing enumeration member 747, 807
 Left enumeration member 804, 1009
 LeftEyePatch enumeration member 805
 LeftFullPalm enumeration member 750, 809
 LeftHypothenar enumeration member 750, 809
 LeftIndex enumeration member 750
 LeftIndexFinger enumeration member 809
 LeftInterdigital enumeration member 750, 809
 LeftLittle enumeration member 750
 LeftLittleFinger enumeration member 809
 LeftLowerPalm enumeration member 750, 809
 LeftMiddle enumeration member 750
 LeftMiddleFinger enumeration member 809
 LeftOther enumeration member 750, 809
 LeftRing enumeration member 750
 LeftRingFinger enumeration member 809
 LeftSlantLoop enumeration member 749, 806
 LeftThenar enumeration member 750, 809
 LeftThumb enumeration member 750, 809
 LeftUpperPalm enumeration member 750, 809
 LeftWritersPalm enumeration member 750, 809
 Licensing 2
 LipMovement enumeration member 1009
 LittleEndian enumeration member 895
 LittleFinger enumeration member 1009
 LivenessCheckFailed enumeration member 752
 LiveScanContactless enumeration member 747, 807
 LiveScanNonOpticalContactlessPlain enumeration member 807
 LiveScanNonOpticalContactlessRolled enumeration member 807
 LiveScanNonOpticalContactPlain enumeration member 807
 LiveScanNonOpticalContactRolled enumeration member 807
 LiveScanOpticalContactlessPlain enumeration member 807
 LiveScanOpticalContactlessRolled enumeration member 807

LiveScanOpticalContactPlain enumeration member 807
 LiveScanOpticalContactRolled enumeration member 807
 LiveScanPalm enumeration member 747, 807
 LiveScanPlain enumeration member 747, 807
 LiveScanRolled enumeration member 747, 807
 LiveScanVerticalSwipe enumeration member 807
 LosslessJpeg class 831
 about LosslessJpeg class 831
 SaveImage 832
 LosslessJpegSaveImageToFile function 202
 LosslessJpegSaveImageToMemory function 203
 LosslessJpegSaveImageToStream function 203

M

MajorCase enumeration member 809
 Male enumeration member 810
 Maroon enumeration member 804
 MatchingFailed enumeration member 746
 Medium enumeration member 753
 MegaMatcher OnCard Structure 2
 MiddleFinger enumeration member 1009
 Moustache enumeration member 805
 MouthOpen enumeration member 805
 MultiColored enumeration member 804
 MultipleBiometrics enumeration member 1009

N

N_64 macro 121
 N_ANSI_C macro 121
 N_BIG_ENDIAN macro 121
 N_BYTE_MAX macro 122
 N_BYTE_MIN macro 122
 N_CALLBACK macro 122
 N_CALLBACK_AW macro 122
 N_CPP macro 122
 N_DEBUG macro 122
 N_DECLARE_HANDLE macro 121
 N_DOUBLE_EPSILON macro 123
 N_DOUBLE_MAX macro 123
 N_DOUBLE_MIN macro 123
 N_E_ARGUMENT macro 78

N_E_ARGUMENT_NULL macro 78	N_LI_PRODUCT_MAX_LENGTH macro 90
N_E_ARGUMENT_OUT_OF_RANGE macro 78	N_LI_TITLE_MAX_LENGTH macro 90
N_E_CLR macro 79	N_LIB macro 126
N_E_COM macro 79	N_LINUX macro 126
N_E_CORE macro 79	N_LONG_MAX macro 126
N_E_END_OF_STREAM macro 79	N_LONG_MIN macro 126
N_E_EXTERNAL macro 79	N_MAC macro 127
N_E_FAILED macro 79	N_MSVC macro 127
N_E_FORMAT macro 80	N_NO_ANSI_FUNC macro 127
N_E_INDEX_OUT_OF_RANGE macro 80	N_NO_INT_64 macro 127
N_E_INVALID_OPERATION macro 80	N_OK macro 82
N_E_IO macro 80	N_PACKED macro 127
N_E_NOT_ACTIVATED macro 80	N_PC_TYPE_ID macro 102
N_E_NOT_IMPLEMENTED macro 81	N_POS_TYPE_MAX macro 128
N_E_NOT_SUPPORTED macro 81	N_POS_TYPE_MIN macro 128
N_E_NULL_REFERENCE macro 81	N_PROCESSOR_INFO_MAX_MODEL_NAME_LENGTH macro 111
N_E_OUT_OF_MEMORY macro 81	N_PROCESSOR_INFO_MAX_VENDOR_NAME_LENGTH macro 111
N_E_OVERFLOW macro 81	N_SBYTE_MAX macro 128
N_E_PARAMETER macro 81	N_SBYTE_MIN macro 128
N_E_PARAMETER_READ_ONLY macro 82	N_SHORT_MAX macro 128
N_E_SYS macro 82	N_SHORT_MIN macro 128
N_E_WIN32 macro 82	N_SINGLE_EPSILON macro 129
N_FAST_FLOAT macro 123	N_SINGLE_MAX macro 129
N_FLOAT_EPSILON macro 123	N_SINGLE_MIN macro 129
N_FLOAT_MAX macro 123	N_SIZE_TYPE_MAX macro 129
N_FLOAT_MIN macro 124	N_SIZE_TYPE_MIN macro 129
N_FUNC_AW macro 124	N_STRUCT_AW macro 129
N_GCC macro 124	N_T macro 130
N_INT_MAX macro 124	N_TYPE_BYTE macro 102
N_INT_MIN macro 124	N_TYPE_BOOL macro 102
N_INT16_MAX macro 125	N_TYPE_CHAR macro 103
N_INT16_MIN macro 125	N_TYPE_DOUBLE macro 103
N_INT32_MAX macro 125	N_TYPE_FLOAT macro 103
N_INT32_MIN macro 125	N_TYPE_INT macro 103
N_INT64_MAX macro 125	N_TYPE_LONG macro 103
N_INT64_MIN macro 125	N_TYPE_SBYTE macro 104
N_INT8_MAX macro 126	N_TYPE_SHORT macro 104
N_INT8_MIN macro 126	N_TYPE_STRING macro 104
N_LI_ACTIVATED_MAX_LENGTH macro 89	N_TYPE_UINT macro 104
N_LI_COMPANY_MAX_LENGTH macro 89	N_TYPE_ULONG macro 104
N_LI_COPYRIGHT_MAX_LENGTH macro 90	

N_TYPE_USHORT macro 104	WriteTo 887
N_UINT_MAX macro 130	NCalcRowSize macro 233
N_UINT_MIN macro 130	NCalcRowSizeEx macro 233
N_UINT16_MAX macro 130	NAlloc function 92
N_UINT16_MIN macro 130	NChar type 117
N_UINT32_MAX macro 131	NClear macro 95
N_UINT32_MIN macro 131	NCompare function 92
N_UINT64_MAX macro 131	NCopy function 93
N_UINT64_MIN macro 131	NCore class 608
N_UINT8_MAX macro 131	about NCore class 608
N_UINT8_MIN macro 131	Alloc 609
N_ULONG_MAX macro 132	CAlloc 609, 610
N_ULONG_MIN macro 132	Clear 610
N_UNICODE macro 132	Compare 610, 611
N_USHORT_MAX macro 132	Copy 611
N_USHORT_MIN macro 132	DllName 609
N_WCHAR_SIZE macro 132	Fill 612
N_WINDOWS macro 133	Free 612
NChar type 116	GetInfo 612
NAlignedFree function 91	Move 613
NAlloc function 91	PtrToArray 613, 614
NationalSemiconductor enumeration member 638	PtrToStructureArray 614
NBiometricStandards class 803	ReAlloc 614, 615
about NBiometricStandards class 803	WriteBufferToStream 615
DllName 803	NCore Library 74, 1014
GetInfo 803	NCore Module 75
NBiometricStandards Library 13, 1011	NCore.h 77
NBiometricStandards Module 73	NCoreGetInfo function 75
NBiometricStandards.h 74	NCoreOnExit function 76
NBiometricStandardsGetInfo function 74	NCoreOnStart function 76
NByte type 116	NCoreOnThreadExit function 76
NBool type 116	NCoreOnThreadStart function 76
NBoolean type 116	NDeviceManager class 822
NBuffer class 886	about NDeviceManager class 822
about NBuffer class 886	DllName 823
Length 887	GetInfo 823
LongLength 887	NDeviceManager Library 136, 1016
NBuffer 886	NDeviceManager Module 151
Ptr 887	NDeviceManager.h 152
ToArray 886	NDeviceManagerGetInfo function 151
UIntPtrLength 887	NDisposable class 615

about NDisposable class 615	Neurotec.Biometrics.Standards.BdifImageSourceType enumeration 811
Dispose 616	Neurotec.Biometrics.Standards.BdifIrisOrientation enumeration 812
NDouble type 117	Neurotec.Biometrics.Standards.BdifIrisScanType enumeration 812
NErrors Module 77	Neurotec.Biometrics.Standards.BdifScaleUnits enumeration 813
NErrors.h 83	Neurotec.Biometrics.Standards.BdifStandard enumeration 813
Neurotec Library 1024	Neurotec.Biometrics.Standards.FmcFeatureHandling enumeration 813
Neurotec namespace 607	Neurotec.Biometrics.Standards.FmcMinutiaOrder enumeration 814
Neurotec.Biometric.Standards 1026	Neurotec.Biometrics.Templates 1027
Neurotec.Biometrics namespace 639	Neurotec.DeviceManager 1028
Neurotec.Biometrics.Extractors 1027	Neurotec.DeviceManager namespace 814
Neurotec.Biometrics.NBiometricType enumeration 746	Neurotec.DeviceManager.FPScannerImageScannedEventHandler type 823
Neurotec.Biometrics.NfeExtractionStatus enumeration 746	Neurotec.DeviceManager.FPScannerManScannerEventHandler type 824
Neurotec.Biometrics.NfeReturnedImage enumeration 747	Neurotec.Images 1028
Neurotec.Biometrics.NfeTemplateSize enumeration 747	Neurotec.Images namespace 824
Neurotec.Biometrics.NFImpressionType enumeration 747	Neurotec.Images.BmpRecording enumeration 884
Neurotec.Biometrics.NFMinutiaFormat enumeration 748	Neurotec.Images.BmpSaveMode enumeration 884
Neurotec.Biometrics.NFMinutiaOrder enumeration 749	Neurotec.Images.NImageRotateFlipType enumeration 885
Neurotec.Biometrics.NFMinutiaType enumeration 749	Neurotec.IO namespace 885
Neurotec.Biometrics.NFPatternClass enumeration 749	Neurotec.IO.NByteOrder enumeration 895
Neurotec.Biometrics.NFPosition enumeration 750	Neurotec.Licensing 1030
Neurotec.Biometrics.NFRidgeCountsType enumeration 752	Neurotec.Licensing namespace 895
Neurotec.Biometrics.NleExtractionStatus enumeration 752	Neurotec.NLibraryGetInfo type 639
Neurotec.Biometrics.NleTemplateSize enumeration 753	Neurotec.NProcessorVendor enumeration 638
Neurotec.Biometrics.Standards namespace 753	Neurotec.SmartCards Library 1030
Neurotec.Biometrics.Standards.BdifEyeColor enumeration 804	Neurotec.SmartCards namespace 899
Neurotec.Biometrics.Standards.BdifEyePosition enumeration 804	Neurotec.SmartCards.ApduClassSecureMessaging enumeration 1001
Neurotec.Biometrics.Standards.BdifFaceExpression enumeration 805	Neurotec.SmartCards.BerTagClass enumeration 1001
Neurotec.Biometrics.Standards.BdifFaceFeaturePointType enumeration 805	Neurotec.SmartCards.BerTagEncoding enumeration 1002
Neurotec.Biometrics.Standards.BdifFaceProperties enumeration 805	Neurotec.SmartCards.Biometry namespace 1002
Neurotec.Biometrics.Standards.BdifFPatternClass enumeration 806	Neurotec.SmartCards.Biometry.SCardBiometricSubtype enumeration 1009
Neurotec.Biometrics.Standards.BdifFPImpressionType enumeration 807	Neurotec.SmartCards.Biometry.SCardBiometricType enumeration 1009
Neurotec.Biometrics.Standards.BdifFPMinutiaType enumeration 808	Neurotec.SmartCards.Biometry.SCardChallengeQualifier enumeration 1010
Neurotec.Biometrics.Standards.BdifFPPosition enumeration 809	
Neurotec.Biometrics.Standards.BdifGender enumeration 810	
Neurotec.Biometrics.Standards.BdifHairColor enumeration 811	

NeurotecException class 616	NFDelta_ structure 571
about NeurotecException class 616	NFDoubleCore structure 572, 647
NeurotecExceptionBase class 616	about NFDoubleCore structure 647
about NeurotecExceptionBase class 616	Y 648
Code 617	NFDoubleCore 647
ManagedStackTrace 617	ToString 647
StackTrace 617	X 648
UnmanagedStackTrace 617	NFDoubleCore_ structure 572
Neutral enumeration member 805	NFE_MODE_ATMEL_FINGERCHIP macro 164
NexGen enumeration member 638	NFE_MODE_AUTHENTEC_AES2501B macro 164
NExtractors class 640	NFE_MODE_AUTHENTEC_AES4000 macro 164
about NExtractors class 640	NFE_MODE_AUTHENTEC_AFS2 macro 164
DllName 641	NFE_MODE_BIOMETRIKA_FX2000 macro 164
GetInfo 641	NFE_MODE_BIOMETRIKA_FX3000 macro 164
NExtractors Library 152, 1016	NFE_MODE_BMF_BLP100 macro 164
NExtractors Module 170	NFE_MODE_CROSSMATCH_VERIFIER300 macro 165
NExtractors.h 171	NFE_MODE_DIGENT_IZZIX macro 165
NExtractorsGetInfo function 171	NFE_MODE_DIGITALPERSONA_UAREU macro 165
NFailed macro 82	NFE_MODE_ETHENTICA macro 168
NFalse macro 133	NFE_MODE_FUJITSU_MBF200 macro 165
NFCore structure 571, 641	NFE_MODE_FUTRONIC_FS80 macro 165
about NFCore structure 641	NFE_MODE_GENERAL macro 165
Angle 642	NFE_MODE_IDENTICATORTECHNOLOGY_DF90 macro 165
Y 643	NFE_MODE_IDENTIX_DFR2090 macro 165
NFCore 641, 642	NFE_MODE_IDENTIX_TOUCHVIEW macro 166
RawAngle 643	NFE_MODE_KEYTRONIC_SECUREDESKTOP macro 166
ToString 642	NFE_MODE_LIGHTTUNING_LTTC500 macro 166
X 643	NFE_MODE_NITGEN_FINGKEY_HAMSTER macro 166
NFCore_ structure 571	NFE_MODE_PRECISEBIOMETRICS_100CS macro 166
NFDelta structure 571, 643	NFE_MODE_SECUGEN_HAMSTER macro 166
about NFDelta structure 643	NFE_MODE_STARTEK_FM200 macro 166
Angle1 645	NFE_MODE_TACOMA_CMOS macro 166
Angle2 645	NFE_MODE_TESTECH_BIOI macro 168
Angle3 646	NFE_MODE_UPEK_TOUCHCHIP macro 167
Y 646	NfeCreate function 158
NFDelta 644	NfeCreateEx function 158
RawAngle1 646	nfeesMatchingFailed enumeration member 161
RawAngle2 646	nfeesQualityCheckFailed enumeration member 161
RawAngle3 646	nfeesTemplateCreated enumeration member 161
ToString 645	nfeesTooFewMinutiae enumeration member 161
X 647	NfeExtract function 155

NfeExtractionStatus enumeration 161	MakeSmallTemplate 657
NfeExtractionStatus_ enumeration 161	MinMinutiaCount 659
NfeGeneralize function 159	Mode 659
NfeGeneralizeEx function 157	ModeAtmelFingerChip 651
NfeIsPalm function 160	ModeAuthentecAes2501B 651
NfeIsTemplateSmall function 156	ModeAuthentecAes4000 652
NfeMakeLightTemplate function 160	ModeAuthentecAFS2 652
NfeMakeSmallTemplate function 156	ModeBiometrikaFX2000 652
NFEP_EXTRACTED_RIDGE_COUNTS macro 167	ModeBiometrikaFX3000 652
NFEP_GENERALIZATION_MAXIMAL_ROTATION macro 167	ModeBmfBlp100 652
NFEP_GENERALIZATION_MIN_MINUTIA_COUNT macro 165	ModeCrossMatchVerifier300 652
NFEP_GENERALIZATION_THRESHOLD macro 167	ModeDigentIzzix 652
NFEP_MIN_MINUTIA_COUNT macro 166	ModeDigitalPersonaUareU 652
NFEP_MODE macro 167	ModeEthentica 652
NFEP_QUALITY_THRESHOLD macro 167	ModeFujitsuMbf200 653
NFEP_RETURNED_IMAGE macro 167	ModeFutronicFS80 653
NFEP_TEMPLATE_SIZE macro 167	ModeGeneral 653
NFEP_USE_QUALITY macro 168	ModelIdenticatorTechnologyDF90 653
NfeReturnedImage enumeration 161	ModelIdentixDfr2090 653
NfeReturnedImage_ enumeration 161	ModelIdentixTouchView 653
nferiBinarized enumeration member 161	ModeKeytronicSecureDesktop 653
nferiNone enumeration member 161	ModeLighTuningLttC500 653
nferiSkeletonized enumeration member 161	ModeNitgenFingkeyHamster 653
NfeTemplateSize enumeration 162	ModePreciseBiometrics100CS 654
NfeTemplateSize_ enumeration 162	ModeSecugenHamster 654
nfetsLarge enumeration member 162	ModeStartekFM200 654
nfetsSmall enumeration member 162	ModeTacomaCmos 654
NfeUpdateTemplate function 157	ModeTestechBioI 654
NFExtractor class 648	ModeUpekTouchChip 654
about NFExtractor class 648	NFExtractor 651
Extract 655	ParameterExtractedRidgeCounts 654
ExtractedRidgeCounts 658	ParameterGeneralizationMaximalRotation 654
GeneralizationMaximalRotation 658	ParameterGeneralizationMinMinutiaCount 654
GeneralizationMinMinutiaCount 659	ParameterGeneralizationThreshold 655
GeneralizationThreshold 659	ParameterMinMinutiaCount 655
Generalize 656	ParameterMode 655
GetStaticParameter 656	ParameterQualityThreshold 655
IsPalm 659	ParameterReturnedImage 655
IsTemplateSmall 657	ParameterTemplateSize 655
MakeLightTemplate 657	ParameterUseQuality 655
	QualityThreshold 660

ReturnedImage 660	RawAngle 664
SetStaticParameter 658	Type 664
TemplateSize 660	ToString 662
UpdateTemplate 658	X 664
UseQuality 660	NFMinutia_ structure 573
NFExtractor Module 153	NFMinutiaFormat enumeration 574
Macros 162	NFMinutiaFormat_ enumeration 574
NFExtractor.h 168	NFMinutiaNeighbor structure 574, 664
NFExtractorParams.h 169	about NFMinutiaNeighbor structure 664
NFill function 93	Empty 665
NFImpressionType enumeration 572	Index 665
NFImpressionType_ enumeration 572	NFMinutiaNeighbor 665
nfitLatentImpression enumeration member 572	RidgeCount 666
nfitLatentLift enumeration member 572	ToString 665
nfitLatentPalmImpression enumeration member 572	NFMinutiaNeighbor_ structure 574
nfitLatentPalmLift enumeration member 572	NFMinutiaOrder enumeration 574
nfitLatentPalmPhoto enumeration member 572	NFMinutiaOrder_ enumeration 574
nfitLatentPalmTracing enumeration member 572	NFMinutiaType enumeration 575
nfitLatentPhoto enumeration member 572	NFMinutiaType_ enumeration 575
nfitLatentTracing enumeration member 572	nfmoAngle enumeration member 574
nfitLiveScanContactless enumeration member 572	nfmoAscending enumeration member 574
nfitLiveScanPalm enumeration member 572	nfmoCartesianYX enumeration member 574
nfitLiveScanPlain enumeration member 572	nfmoCartesianXY enumeration member 574
nfitLiveScanRolled enumeration member 572	nfmoDescending enumeration member 574
nfitNonliveScanPalm enumeration member 572	nfmoPolar enumeration member 574
nfitNonliveScanPlain enumeration member 572	nfmtBifurcation enumeration member 575
nfitNonliveScanRolled enumeration member 572	nfmtEnd enumeration member 575
nfitSwipe enumeration member 572	nfmtOther enumeration member 575
NFloat type 117	nfmtUnknown enumeration member 575
nfmfHasCurvature enumeration member 574	NFPatternClass enumeration 575
nfmfHasG enumeration member 574	NFPatternClass_ enumeration 575
nfmfHasQuality enumeration member 574	nfpcAccidentalWhorl enumeration member 575
nfmfNone enumeration member 574	nfpcAmputation enumeration member 575
NFMinutia structure 573, 660	nfpcCentralPocketLoop enumeration member 575
about NFMinutia structure 660	nfpcDoubleLoop enumeration member 575
Angle 662	nfpcLeftSlantLoop enumeration member 575
Curvature 663	nfpcPlainArch enumeration member 575
G 663	nfpcPlainWhorl enumeration member 575
Y 663	nfpcRadialLoop enumeration member 575
NFMinutia 661, 662	nfpcRightSlantLoop enumeration member 575
Quality 663	nfpcScar enumeration member 575

nfpCtentedArch enumeration member 575	NFR_MAX_DIMENSION macro 579
nfpCtlnarLoop enumeration member 575	NFR_MAX_DOUBLE_CORE_COUNT macro 579
nfpCtUnknown enumeration member 575	NFR_MAX_FINGER_CORE_COUNT macro 579
nfpCtWhorl enumeration member 575	NFR_MAX_FINGER_DELTA_COUNT macro 579
nfpLeftFullPalm enumeration member 576	NFR_MAX_FINGER_DIMENSION macro 582
nfpLeftHypothenar enumeration member 576	NFR_MAX_FINGER_DOUBLE_CORE_COUNT macro 582
nfpLeftIndex enumeration member 576	NFR_MAX_FINGER_MINUTIA_COUNT macro 582
nfpLeftInterdigital enumeration member 576	NFR_MAX_MINUTIA_COUNT macro 579
nfpLeftLittle enumeration member 576	NFR_MAX_PALM_CORE_COUNT macro 582
nfpLeftLowerPalm enumeration member 576	NFR_MAX_PALM_DELTA_COUNT macro 582
nfpLeftMiddle enumeration member 576	NFR_MAX_PALM_DIMENSION macro 582
nfpLeftOther enumeration member 576	NFR_MAX_PALM_DOUBLE_CORE_COUNT macro 583
nfpLeftRing enumeration member 576	NFR_MAX_PALM_MINUTIA_COUNT macro 583
nfpLeftThenar enumeration member 576	NFR_RESOLUTION macro 579
nfpLeftThumb enumeration member 576	NFR_SAVE_BLOCKED_ORIENTS macro 580
nfpLeftUpperPalm enumeration member 576	NFR_SAVE_V2 macro 581
nfpLeftWritersPalm enumeration member 576	NFR_SAVE_V3 macro 581
NFPosition enumeration 576	NFR_SKIP_BLOCKED_ORIENTS macro 580
NFPosition_ enumeration 576	NFR_SKIP_CURVATURES macro 580
nfpPlainLeftFourFingers enumeration member 576	NFR_SKIP_GS macro 580
nfpPlainLeftThumb enumeration member 576	NFR_SKIP_QUALITIES macro 580
nfpPlainRightFourFingers enumeration member 576	NFR_SKIP RIDGE_COUNTS macro 580
nfpPlainRightThumb enumeration member 576	NFR_SKIP_SINGULAR_POINTS macro 581
nfpPlainThumbs enumeration member 576	nfrctEightNeighbors enumeration member 578
nfpRightFullPalm enumeration member 576	nfrctEightNeighborsWithIndexes enumeration member 578
nfpRightHypothenar enumeration member 576	nfrctFourNeighbors enumeration member 578
nfpRightIndex enumeration member 576	nfrctFourNeighborsWithIndexes enumeration member 578
nfpRightInterdigital enumeration member 576	nfrctNone enumeration member 578
nfpRightLittle enumeration member 576	nfrctUnspecified enumeration member 578
nfpRightLowerPalm enumeration member 576	NFRecord class 666
nfpRightMiddle enumeration member 576	about NFRecord class 666
nfpRightOther enumeration member 576	CbeffProductType 691
nfpRightRing enumeration member 576	Check 676
nfpRightThenar enumeration member 576	Clone 677
nfpRightThumb enumeration member 576	Cores 692
nfpRightUpperPalm enumeration member 576	Deltas 692
nfpRightWritersPalm enumeration member 576	DoubleCores 692
nfpUnknown enumeration member 576	FlagSaveBlockedOrient 673
nfpUnknownPalm enumeration member 576	FlagSaveV2 673
NFR_MAX_CORE_COUNT macro 581	FlagSaveV3 673
NFR_MAX_DELTA_COUNT macro 581	FlagSkipBlockedOrient 674

FlagSkipCurvatures 674	MinutiaFormat 693
FlagSkipGs 674	NFRecord 669, 670, 671, 672
FlagSkipQualities 674	PatternClass 693
FlagSkipRidgeCounts 674	Position 694
FlagSkipSingularPoints 674	Quality 694
FromHandle 677	Resolution 676
G 692	RidgeCountsType 694
GetCbeffProductType 677, 678	Save 687, 688
GetG 678	SaveV1 689, 690
GetHeight 679	SortMinutiae 691
GetHorzResolution 679, 680	TruncateMinutiae 691
GetImpressionType 680	TruncateMinutiaeByQuality 691
GetMaxSize 681	VertResolution 694
GetMaxSizeV1 681, 682	Width 694
GetPatternClass 682, 683	NFRecord Module 518
GetPosition 683	NFRecord.CoreCollection class 672
GetQuality 684	about NFRecord.CoreCollection class 672
GetSize 684, 685	NFRecord.DeltaCollection class 673
GetSizeV1 685	about NFRecord.DeltaCollection class 673
GetVertResolution 685, 686	NFRecord.DoubleCoreCollection class 673
GetWidth 686	about NFRecord.DoubleCoreCollection class 673
Height 692	NFRecord.h 583
HorzResolution 692	NFRecord.MinutiaCollection class 673
ImpressionType 693	about NFRecord.MinutiaCollection class 673
MaxCoreCount 674	NFRecord.MinutiaNeighborsCollection class 673
MaxDeltaCount 674	about NFRecord.MinutiaNeighborsCollection class 673
MaxDimension 674	NFRecordAddCore function 523
MaxDoubleCoreCount 675	NFRecordAddDelta function 525
MaxFingerCoreCount 675	NFRecordAddDoubleCore function 526
MaxFingerDeltaCount 675	NFRecordAddMinutia function 526
MaxFingerDimension 675	NFRecordCheck function 527
MaxFingerDoubleCoreCount 675	NFRecordClearCores function 527
MaxFingerMinutiaCount 675	NFRecordClearDeltas function 527
MaxMinutiaCount 675	NFRecordClearDoubleCores function 528
MaxPalmCoreCount 675	NFRecordClearMinutiae function 528
MaxPalmDeltaCount 676	NFRecordClone function 529
MaxPalmDimension 676	NFRecordCreate function 529
MaxPalmDoubleCoreCount 676	NFRecordCreateEx function 530
MaxPalmMinutiaCount 676	NFRecordCreateFromMemory function 531
Minutiae 693	NFRecordGetCbeffProductType function 532
MinutiaeNeighbors 693	NFRecordGetCbeffProductTypeMem function 532

NFRecordGetCore function 533	NFRecordGetWidth function 554
NFRecordGetCoreCapacity function 533	NFRecordGetWidthMem function 554
NFRecordGetCoreCount function 534	NFRecordInfo class 694
NFRecordGetCores function 535	about NFRecordInfo class 694
NFRecordGetDelta function 535	HeaderSize 695
NFRecordGetDeltaCapacity function 536	MajorVersion 695
NFRecordGetDeltaCount function 536	MinorVersion 695
NFRecordGetDeltas function 537	Size 695
NFRecordGetDoubleCore function 537	NFRecordInfo structure 578
NFRecordGetDoubleCoreCapacity function 538	NFRecordInfo_ structure 578
NFRecordGetDoubleCoreCount function 538	NFRecordInfoDispose function 555
NFRecordGetDoubleCores function 539	NFRecordInsertCore function 555
NFRecordGetG function 540	NFRecordInsertDelta function 556
NFRecordGetGMem function 540	NFRecordInsertDoubleCore function 556
NFRecordGetHeight function 541	NFRecordInsertMinutia function 557
NFRecordGetHeightMem function 541	NFRecordRemoveCore function 558
NFRecordGetHorzResolution function 542	NFRecordRemoveDelta function 558
NFRecordGetHorzResolutionMem function 542	NFRecordRemoveDoubleCore function 559
NFRecordGetImpressionType function 543	NFRecordRemoveMinutia function 559
NFRecordGetImpressionTypeMem function 543	NFRecordSaveToMemory function 560
NFRecordGetMaxSize function 524	NFRecordSaveToMemoryV1 function 517
NFRecordGetMaxSizeV1 function 515	NFRecordSetCbeffProductType function 560
NFRecordGetMinutia function 544	NFRecordSetCore function 561
NFRecordGetMinutiaCapacity function 544	NFRecordSetCoreCapacity function 561
NFRecordGetMinutiaCount function 545	NFRecordSetDelta function 562
NFRecordGetMinutiae function 546	NFRecordSetDeltaCapacity function 563
NFRecordGetMinutiaFormat function 546	NFRecordSetDoubleCore function 563
NFRecordGetMinutiaNeighbor function 547	NFRecordSetDoubleCoreCapacity function 564
NFRecordGetMinutiaNeighborCount function 547	NFRecordSetG function 564
NFRecordGetMinutiaNeighbors function 548	NFRecordSetImpressionType function 565
NFRecordGetPatternClass function 548	NFRecordSetMinutia function 565
NFRecordGetPatternClassMem function 549	NFRecordSetMinutiaCapacity function 566
NFRecordGetPosition function 550	NFRecordSetMinutiaFormat function 566
NFRecordGetPositionMem function 550	NFRecordSetMinutiaNeighbor function 567
NFRecordGetQuality function 551	NFRecordSetPatternClass function 567
NFRecordGetQualityMem function 551	NFRecordSetPosition function 568
NFRecordGetRidgeCountsType function 552	NFRecordSetQuality function 568
NFRecordGetSize function 552	NFRecordSetRidgeCountsType function 569
NFRecordGetSizeV1 function 516	NFRecordSortMinutiae function 569
NFRecordGetVertResolution function 553	NFRecordTruncateMinutiae function 570
NFRecordGetVertResolutionMem function 553	NFRecordTruncateMinutiaeByQuality function 570

- NFRecordV1 Module 515
- NFRecordV1.h 518
- NFree function 94
- NFRidgeCountsType enumeration 578
- NFRidgeCountsType_ enumeration 578
- NFT_MAX_RECORD_COUNT macro 604
- NFT_PROCESS_FIRST_RECORD_ONLY macro 604
- NFTemplate class 695
 - about NFTemplate class 695
 - CalculateSize 703
 - Check 703, 704
 - Clone 704
 - FlagProcessFirstRecordOnly 703
 - FromHandle 704
 - GetRecordCount 705
 - GetSize 705, 706
 - IsPalm 712
 - IsPalmTemplate 706
 - MaxRecordCount 703
 - NFTemplate 697, 698, 699, 700
 - Pack 707
 - Records 712
 - Save 708, 709
 - Unpack 710, 711
- NFTemplate Module 588
- NFTemplate.h 604
- NFTemplate.RecordCollection class 701
 - about NFTemplate.RecordCollection class 701
 - Add 701, 702
 - AddCopy 702
- NFTemplateAddRecord function 589
- NFTemplateAddRecordCopy function 590
- NFTemplateAddRecordFromMemory function 591
- NFTemplateCalculateSize function 592
- NFTemplateCheck function 592
- NFTemplateClearRecords function 593
- NFTemplateClone function 593
- NFTemplateCreate function 594
- NFTemplateCreateEx function 594
- NFTemplateCreateFromMemory function 595
- NFTemplateGetRecord function 596
- NFTemplateGetRecordCapacity function 596
- NFTemplateGetRecordCount function 597
- NFTemplateGetRecordCountMem function 597
- NFTemplateGetSize function 598
- NFTemplateInfo class 712
 - about NFTemplateInfo class 712
 - HeaderSize 713
 - MajorVersion 713
 - MinorVersion 713
 - RecordInfos 713
 - Size 713
- NFTemplateInfo.RecordInfoCollection class 713
 - about NFTemplateInfo.RecordInfoCollection class 713
- NFTemplateInfoDispose function 598
- NFTemplateIsPalm function 599
- NFTemplateIsPalmMem function 599
- NFTemplatePack function 600
- NFTemplateRemoveRecord function 601
- NFTemplateSaveToMemory function 601
- NFTemplateSetRecordCapacity function 602
- NFTemplateUnpack function 603
- NGeometry Module 84
- NGeometry.h 87
- NGrayscaleImage class 832
 - about NGrayscaleImage class 832
 - this 834
- NGrayscaleImage Module 204
- NGrayscaleImage.h 205
- NHandle type 117
- NImage class 834
 - about NImage class 834
 - Clone 835
 - Copy 836
 - Create 836, 837
 - Crop 837
 - FlipDiagonally 838
 - FlipHorizontally 838
 - FlipVertically 838
 - FromBitmap 838
 - FromData 838, 839, 840, 841, 842
 - FromFile 843

- FromHandle 844
- FromHBitmap 844
- FromImage 844, 845, 846, 847, 848
- GetWrapper 848, 849, 850, 851, 852, 853
- Height 855
- HorzResolution 855
- LongSize 855
- LongStride 855
- PixelFormat 856
- Pixels 856
- RotateFlip 854
- Save 854
- Size 856
- Stride 856
- ToBitmap 854
- ToHBitmap 854
- VertResolution 857
- Width 857
- NImage Module 205
- NImage.h 222
- NImageClone function 206
- NImageCopy function 206
- NImageCreate function 207
- NImageCreateFromData function 208
- NImageCreateFromDataPart function 209
- NImageCreateFromFile function 210
- NImageCreateFromImage function 211
- NImageCreateFromImageEx function 212
- NImageCreateWrapper function 213
- NImageCrop function 214
- NImageFile class 857
 - about NImageFile class 857
 - Close 858
 - Format 859
 - FromFile 858
 - ReadImage 858
- NImageFile Module 223
- NImageFile.h 226
- NImageFileClose function 223
- NImageFileCreate function 224
- NImageFileGetFormat function 224
- NImageFileIsOpened function 225
- NImageFileReadImage function 225
- NImageFlipDiagonally function 214
- NImageFlipHorizontally function 215
- NImageFlipVertically function 215
- NImageFormat class 859
 - about NImageFormat class 859
 - Bmp 861
 - CanRead 866
 - CanWrite 866
 - CanWriteMultiple 866
 - DefaultFileExtension 866
 - FileFilter 866
 - Formats 861
 - Gif 861
 - IHead 861
 - Jpeg 861
 - Jpeg2K 861
 - LoadImage 862
 - Name 866
 - OpenFile 863, 864
 - Png 861
 - SaveImage 864
 - SaveImages 865
 - Select 865
 - Tiff 861
 - Wsq 861
- NImageFormat Module 244
- NImageFormat.h 259
- NImageFormat.ImageFormatCollection class 860
 - about NImageFormat.ImageFormatCollection class 860
- NImageFormatCanRead function 245
- NImageFormatCanWrite function 246
- NImageFormatCanWriteMultiple function 246
- NImageFormatGetBmp function 247
- NImageFormatGetDefaultFileExtension function 247
- NImageFormatGetFileFilter function 248
- NImageFormatGetFormat function 248
- NImageFormatGetFormatCount function 249
- NImageFormatGetIHead function 249
- NImageFormatGetJpeg function 249

- NImageFormatGetJpeg2K function 250
- NImageFormatGetName function 250
- NImageFormatGetPng function 251
- NImageFormatGetTiff function 252
- NImageFormatGetWsq function 252
- NImageFormatLoadImageFromFile function 253
- NImageFormatLoadImageFromMemory function 254
- NImageFormatLoadImageFromStream function 254
- NImageFormatOpenFile function 251
- NImageFormatOpenFileFromMemory function 255
- NImageFormatOpenFileFromStream function 256
- NImageFormatSaveImagesToFile function 253
- NImageFormatSaveImageToFile function 256
- NImageFormatSaveImageToMemory function 257
- NImageFormatSaveImageToStream function 257
- NImageFormatSelect function 258
- NImageGetHeight function 216
- NImageGetHorzResolution function 216
- NImageGetPixelFormat function 216
- NImageGetPixels function 217
- NImageGetSize function 217
- NImageGetStride function 218
- NImageGetVertResolution function 219
- NImageGetWidth function 219
- NImageRotateFlip function 219
- NImageRotateFlipType enumeration 221
- NImageRotateFlipType_ enumeration 221
- NImages class 867
 - about NImages class 867
 - DllName 867
 - GetGrayscaleColorWrapper 867
 - GetInfo 868
 - GetOpenFileFilter 868
 - GetOpenFileFilterString 868
 - GetSaveFileFilter 868
 - GetSaveFileFilterString 868
- NImages Library 188, 1019
- NImages Module 227
- NImages.h 228
- NImageSaveToFile function 220
- NImagesGetGrayscaleColorWrapperEx function 227
- NImagesGetInfo function 228
- NIndexPair structure 115, 617
 - about NIndexPair structure 617
 - Index1 618
 - Index2 618
 - NIndexPair 618
 - ToString 618
- NIndexPair_ structure 115
- NInfo class 619
 - about NInfo class 619
- NInfoObject class 619
 - about NInfoObject class 619
- NInt type 117
- NInt16 type 117
- NInt32 type 118
- NInt64 type 118
- NInt8 type 118
- nirftFlipY enumeration member 221
- nirftFlipNone enumeration member 221
- nirftFlipX enumeration member 221
- nirftFlipXY enumeration member 221
- nirftNone enumeration member 221
- nirftRotate180 enumeration member 221
- nirftRotate270 enumeration member 221
- nirftRotate90 enumeration member 221
- nirftRotateNone enumeration member 221
- NleCompress function 178
- NleCompressEx function 173
- NleCreate function 179
- NleDetectFace function 174
- NleDetectFaces function 174
- NleDetectFacialFeatures function 175
- NleDetectionDetails structure 182, 714
 - about NleDetectionDetails structure 714
 - Eyes 714
 - EyesAvailable 714
 - Face 714
 - FaceAvailable 714
- NleDetectionDetails_ structure 182
- NleEyes structure 182, 714
 - about NleEyes structure 714

First 715	about NleRotation structure 716
FirstConfidence 715	Yaw 717
Second 715	Pitch 717
SecondConfidence 715	Roll 717
NleEyes_ structure 182	NleRotation_ structure 181
nleesEyesNotDetected enumeration member 182	NleTemplateSize enumeration 181
nleesFaceNotDetected enumeration member 182	NleTemplateSize_ enumeration 181
nleesFaceTooCloseToImageBorder enumeration member 182	nletsLarge enumeration member 181
nleesGeneralizationFailed enumeration member 182	nletsMedium enumeration member 181
nleesLivenessCheckFailed enumeration member 182	nletsSmall enumeration member 181
nleesNone enumeration member 182	NLExtractor class 717
nleesQualityCheckExposureFailed enumeration member 182	about NLExtractor class 717
nleesQualityCheckGrayscaleDensityFailed enumeration member 182	Compress 721
nleesQualityCheckSharpnessFailed enumeration member 182	DetectFace 721
nleesTemplateCreated enumeration member 182	DetectFaces 722
NleExtract function 175	DetectFacialFeatures 722
NleExtractionStatus enumeration 182	Extract 722
NleExtractionStatus_ enumeration 182	ExtractNext 723
NleExtractNext function 176	ExtractStart 723
NleExtractNextEx function 177	ExtractUsingDetails 723
NleExtractStart function 179	FaceConfidenceThreshold 725
NleExtractUsingDetails function 177	FaceQualityThreshold 725
NleFace structure 183, 715	FavorLargestFace 725
about NleFace structure 715	Generalize 724
Confidence 716	GetStaticParameter 724
Rectangle 716	LivenessThreshold 725
Rotation 716	MaxIod 725
NleFace_ structure 183	MaxRecordsPerTemplate 726
NleGeneralize function 180	MaxRollAngleDeviation 726
NleGeneralizeEx function 178	MinIod 726
NLEP_FACE_CONFIDENCE_THRESHOLD macro 184	NLExtractor 719
NLEP_FACE_QUALITY_THRESHOLD macro 185	ParameterFaceConfidenceThreshold 719
NLEP_FAVOR_LARGEST_FACE macro 185	ParameterFaceQualityThreshold 719
NLEP_LIVENESS_THRESHOLD macro 185	ParameterFavorLargestFace 720
NLEP_MAX_IOD macro 185	ParameterLivenessThreshold 720
NLEP_MAX_RECORDS_PER_TEMPLATE macro 186	ParameterMaxIod 720
NLEP_MAX_ROLL_ANGLE_DEVIATION macro 184	ParameterMaxRecordsPerTemplate 720
NLEP_MIN_IOD macro 186	ParameterMaxRollAngleDeviation 720
NLEP_TEMPLATE_SIZE macro 185	ParameterMinIod 720
NLEP_USE_LIVENESS_CHECK macro 186	ParameterTemplateSize 720
NleRotation structure 181, 716	ParameterUseLivenessCheck 721

- SetStaticParameter 724
- TemplateSize 726
- UseLivenessCheck 726
- NLExtractor Module 171
 - Macros 183
- NLExtractor.h 186
- NLExtractorParams.h 187
- NLibraryInfo class 619
 - about NLibraryInfo class 619
 - Activated 620
 - Company 620
 - Copyright 620
 - Product 620
 - Retrieve 620
 - Title 620
 - Version 620
- NLibraryInfo Module 88
- NLibraryInfo.h 90
- NLibraryInfo_ structure 89
- NLicense class 895
 - about NLicense class 895
 - GetInfo 896
 - IsComponentActivated 896
 - Obtain 896
 - Release 897
- NLicenseGetInfo function 262
- NLicenseInfo class 897
 - about NLicenseInfo class 897
 - DistributorId 897
 - IsObtained 898
 - SerialNumber 898
- NLicenseInfo structure 267
- NLicenseInfo_ structure 267
- NLicenseIsComponentActivated function 262
- NLicenseObtain function 263
- NLicenseRelease function 265
- NLicensing class 898
 - about NLicensing class 898
 - DllName 898
 - GetInfo 898
- NLicensing Library 260, 1022
- NLicensing Module 261
- NLicensing.h 267
- NLicensingGetInfo function 266
- NLong type 118
- NMemory Module 91
- NMemory.h 95
- NMemoryStream class 887
 - about NMemoryStream class 887
 - Capacity 889
 - NMemoryStream 888, 889
 - ToArray 889
 - WriteTo 889
- NMonochromeImage class 869
 - about NMonochromeImage class 869
 - this 870
- NMonochromeImage Module 228
- NMonochromeImage.h 230
- NMonochromeImageGetPixel function 229
- NMonochromeImageSetPixel function 229
- NMove function 94
- NObject class 621
 - about NObject class 621
 - CopyParameters 621
 - Free 621
 - GetNativeType 622
 - GetParameter 622
 - Handle 623
 - Owner 623
 - Reset 622
 - SetParameter 623
- NObject Module 96
- NObject.h 101
- NObjectCopyParameters function 96
- NObjectFree function 97
- NObjectGetOwner function 97
- NObjectGetParameter function 97
- NObjectGetParameterWithPart function 98
- NObjectGetType function 98
- NObjectReset function 99
- NObjectSetParameter function 99
- NObjectSetParameterWithPart function 99

- NObjectTypeOf function 100
- NoEar enumeration member 805
- None enumeration member 747, 748, 752, 813, 814, 885, 1001, 1009
- NonliveScanPalm enumeration member 747, 807
- NonliveScanPlain enumeration member 747, 807
- NonliveScanRolled enumeration member 747, 807
- NotActivatedException class 623
 - about NotActivatedException class 623
- NotSpecified enumeration member 805
- NParameterMakeld macro 105
- NParameters Module 101
- NParameters.h 105
- npfGrayscale enumeration member 232
- npfMonochrome enumeration member 232
- npfRgb enumeration member 232
- NPixelFormat Module 230
- NPixelFormat structure 870
 - != 875
 - == 875
 - about NPixelFormat structure 870
 - BitsPerPixel 876
 - CalcRowLongSize 872
 - CalcRowSize 872, 873
 - Equals 873
 - GetHashCode 873
 - GetRowLongSize 873, 874
 - GetRowSize 874
 - Grayscale 871
 - IsValid 875
 - Monochrome 871
 - NPixelFormat 876
 - Rgb 872
 - ToString 875
 - uint 876
- NPixelFormat type 232
- NPixelFormat.h 234
- NPixelFormat_ enumeration 232
- NPixelFormatGetBitsPerPixel macro 233
- NPixelFormatGetRowSize macro 233
- NPixelFormatGetRowSizeEx macro 233
- NPixelFormatIsValid function 231
- NPoint structure 84
- NPoint_ structure 84
- NPointD structure 85
- NPointD_ structure 85
- NPointF structure 86
- NPointF_ structure 86
- NPosType type 118
- NProcessorInfo class 624
 - about NProcessorInfo class 624
 - GetModelInfo 625
 - Is3DNowSupported 625
 - IsLZCntSupported 625
 - IsMmxSupported 625
 - IsPopCntSupported 625
 - IsSse2Supported 625
 - IsSse3Supported 625
 - IsSse41Supported 625
 - IsSse42Supported 626
 - IsSse4aSupported 626
 - IsSse5Supported 626
 - IsSseSupported 626
 - IsSsse3Supported 626
 - ModelName 626
 - Vendor 626
 - VendorName 626
- NProcessorInfo Module 106
- NProcessorInfo.h 111
- NProcessorInfoGetModelNameA function 106
- NProcessorInfoGetModelNameW function 107
- NProcessorInfoGetVendor function 107
- NProcessorInfoGetVendorNameA function 108
- NProcessorInfoGetVendorNameW function 108
- NProcessorInfoIs3DNowSupported function 109
- NProcessorInfoIsMmxSupported function 109
- NProcessorInfoIsSse2Supported function 109
- NProcessorInfoIsSse3Supported function 110
- NProcessorInfoIsSseSupported function 110
- NProcessorVendor enumeration 110
- NProcessorVendor_ enumeration 110
- npvAmd enumeration member 110
- npvCentaur enumeration member 110

npvCyril enumeration member 110	EDriveNotFound 631
npvIntel enumeration member 110	EEndOfStream 631
npvNationalSemiconductor enumeration member 110	EExternal 631
npvNexGen enumeration member 110	EFailed 631
npvRiseTechnology enumeration member 110	EFileLoad 632
npvSiS enumeration member 110	EFileNotFound 632
npvTransmeta enumeration member 110	EFormat 632
npvUmc enumeration member 110	EIndexOutOfRange 632
npvUnknown enumeration member 110	EInvalidCast 632
NRange structure 626	EInvalidEnumArgument 632
about NRange structure 626	EInvalidOperation 632
From 627	EIO 632
NRange 627	ENotActivated 632
To 627	ENotImplemented 633
ToString 627	ENotSupported 633
NRational structure 115, 628	ENullReference 633
about NRational structure 628	EOutOfMemory 633
Denominator 629	EOverflow 633
Empty 628	EParameter 633
NRational 628	EParameterReadOnly 633
Numerator 629	EPathTooLong 633
ToString 629	ESecurity 633
NRational_ structure 115	ESys 634
NReAlloc function 95	EWin32 634
NRect structure 85	IsFailed 634
NRect_ structure 85	IsSucceeded 634
NRectD structure 86	Ok 634
NRectD_ structure 86	RaiseError 635
NRectF structure 86	SetError 635
NRectF_ structure 86	NResult type 118
NResult class 629	NRgb structure 232, 876
about NResult class 629	about NRgb structure 876
Check 634	Blue 877
EArgument 630	Green 877
EArgumentNull 630	NRgb 877
EArgumentOutOfRange 630	Red 878
EArithmetic 631	ToString 877
EClr 631	NRgb_ structure 232
ECom 631	NRgbConst macro 234
ECore 631	NRgbImage class 878
EDirectoryNotFound 631	about NRgbImage class 878

- this 879
- NRgbImage Module 235
- NRgbImage.h 236
- NRgbImageGetPixel function 235
- NRgbImageSetPixel function 236
- NSByte type 119
- NShort type 119
- NSingle type 119
- NSize structure 85
- NSize_ structure 85
- NSizeD structure 87
- NSizeD_ structure 87
- NSizeF structure 87
- NSizeF_ structure 87
- NSizeType type 119
- NSmartCards Library 268, 1022
- NSmartCardsBiometry Module 307
- NSmartCardsBiometry.h 318
- NSmartCardsCommands Module 320
- NSmartCardsCommands.h 409
- NSmartCardsDataElements Module 459
- NSmartCardsDataElements.h 493
- NStream class 889
 - about NStream class 889
 - CanRead 894
 - CanSeek 894
 - CanWrite 894
 - Close 890
 - Flush 890
 - FromHandle 890, 891
 - FromStream 891
 - Handle 894
 - Length 894
 - Position 894
 - Read 891, 892
 - ReadByte 892
 - Seek 892
 - SetLength 893
 - Write 893
 - WriteByte 894
- NSucceeded macro 83
- NTemplate class 726
 - about NTemplate class 726
 - AddFaces 730, 731
 - AddFacesCopy 731
 - AddFingers 732
 - AddFingersCopy 733
 - AddIrises 733, 734
 - AddIrisesCopy 734
 - AddPalms 734, 735
 - AddPalmsCopy 735
 - CalculateSize 735
 - Check 736
 - Clear 736
 - Clone 736
 - Faces 743
 - Fingers 743
 - FromHandle 737
 - GetSize 737
 - Irises 744
 - NTemplate 728, 729, 730
 - Pack 738
 - Palms 744
 - RemoveFaces 739
 - RemoveFingers 739
 - RemoveIrises 739
 - RemovePalms 739
 - Save 739, 740, 741
 - Unpack 741, 742, 743
- NTemplateInfo class 744
 - about NTemplateInfo class 744
 - FacesInfo 745
 - FingersInfo 745
 - HeaderSize 745
 - IrisesInfo 745
 - MajorVersion 745
 - MinorVersion 745
 - PalmsInfo 745
 - Size 745
- NTemplates class 745
 - about NTemplates class 745
 - DllName 746

- GetInfo 746
- NTemplates Library 514, 1022
- NTemplates Module 605
- NTemplates.h 606
- NTemplatesGetInfo function 606
- NTypeGetBaseType function 100
- NTypeGetName function 101
- NTypes Module 112
- NTypes.h 133
- NTrue macro 133
- NUInt type 119
- NUInt16 type 120
- NUInt32 type 120
- NUInt64 type 120
- NUInt8 type 120
- NULL macro 133
- NULong type 120
- NURational structure 115, 635
 - about NURational structure 635
 - Denominator 636
 - Empty 636
 - Numerator 636
 - NURational 635
 - ToString 636
- NURational_ structure 115
- NUShort type 120
- NWChar type 121

O

- On-card Biometric APIs 6
- On-Card Biometric Components 4
- Orange enumeration member 811
- Other enumeration member 749, 807, 808
- Overview 4

P

- PalmPrint enumeration member 1009
- ParameterException class 636
 - about ParameterException class 636
- ParameterReadOnlyException class 637
 - about ParameterReadOnlyException class 637

- PC Part 5, 9
- Pink enumeration member 804, 811
- PixelsPerCentimeter enumeration member 813
- PixelsPerInch enumeration member 813
- PlainArch enumeration member 749, 806
- PlainLeftFourFingers enumeration member 750, 809
- PlainLeftThumb enumeration member 750, 809
- PlainRightFourFingers enumeration member 750, 809
- PlainRightThumb enumeration member 750, 809
- PlainThumbs enumeration member 750, 809
- PlainWhorl enumeration member 749, 806
- Png class 880
 - about Png class 880
 - DefaultCompressionLevel 880
 - LoadImage 880, 881
 - SaveImage 881, 882
- Png Module 237
- Png.h 241
- PNG_DEFAULT_COMPRESSION_LEVEL macro 241
- PngLoadImageFromFile function 237
- PngLoadImageFromMemory function 238
- PngLoadImageFromStream function 238
- PngSaveImageToFile function 239
- PngSaveImageToMemory function 239
- PngSaveImageToStream function 240
- Point2D enumeration member 805
- PointerFinger enumeration member 1009
- Polar enumeration member 749, 814
- Primitive enumeration member 1002
- PrimitiveBerTlv class 967
 - about PrimitiveBerTlv class 967
 - GetValue 968
 - GetValueAsByte 968
 - GetValueAsBoolean 969
 - GetValueAsInt16 969
 - GetValueAsInt32 969
 - GetValueAsInteger 969
 - GetValueAsSByte 969
 - GetValueAsUInt16 970
 - GetValueAsUInt32 970
 - SetLength 970

SetValue 970, 971, 972

SetValueAsInteger 972

ValuePtr 972

PrimitiveBerTlvGetValue function 287

PrimitiveBerTlvGetValueAsByte function 287

PrimitiveBerTlvGetValueAsBoolean function 288

PrimitiveBerTlvGetValueAsInt16 function 288

PrimitiveBerTlvGetValueAsInt32 function 289

PrimitiveBerTlvGetValueAsInteger function 289

PrimitiveBerTlvGetValueAsSByte function 290

PrimitiveBerTlvGetValueAsUInt16 function 290

PrimitiveBerTlvGetValueAsUInt32 function 291

PrimitiveBerTlvGetValuePtr function 291

PrimitiveBerTlvSetLength function 292

PrimitiveBerTlvSetValue function 292

PrimitiveBerTlvSetValueAsByte function 293

PrimitiveBerTlvSetValueAsBoolean function 293

PrimitiveBerTlvSetValueAsInt16 function 294

PrimitiveBerTlvSetValueAsInt32 function 294

PrimitiveBerTlvSetValueAsInteger function 295

PrimitiveBerTlvSetValueAsSByte function 295

PrimitiveBerTlvSetValueAsUInt16 function 296

PrimitiveBerTlvSetValueAsUInt32 function 296

Private enumeration member 1001

Progressive enumeration member 812

Proprietary enumeration member 1001

Purple enumeration member 811

Q

QualityCheckExposureFailed enumeration member 752

QualityCheckFailed enumeration member 746

QualityCheckGrayscaleDensityFailed enumeration member 752

QualityCheckSharpnessFailed enumeration member 752

R

RadialLoop enumeration member 749, 806

RaisedBrows enumeration member 805

Red enumeration member 811

Retina enumeration member 1009

RidgeCounts enumeration member 813

Right enumeration member 804, 1009

RightEyePatch enumeration member 805

RightFullPalm enumeration member 750, 809

RightHypothenar enumeration member 750, 809

RightIndex enumeration member 750

RightIndexFinger enumeration member 809

RightInterdigital enumeration member 750, 809

RightLittle enumeration member 750

RightLittleFinger enumeration member 809

RightLowerPalm enumeration member 750, 809

RightMiddle enumeration member 750

RightMiddleFinger enumeration member 809

RightOther enumeration member 750, 809

RightRing enumeration member 750

RightRingFinger enumeration member 809

RightSlantLoop enumeration member 749, 806

RightThenar enumeration member 750, 809

RightThumb enumeration member 750, 809

RightUpperPalm enumeration member 750, 809

RightWritersPalm enumeration member 750, 809

RingFinger enumeration member 1009

RiseTechnology enumeration member 638

Rotate180 enumeration member 885

Rotate270 enumeration member 885

Rotate90 enumeration member 885

RotateNone enumeration member 885

S

Samples 11

Sandy enumeration member 811

ScannedPhoto enumeration member 811

Scar enumeration member 749, 806

SCARD_BDT_CT_TAG_CHALLENGE macro 312

SCARD_BDT_CT_TAG_CHALLENGE_QUALIFIER macro 312

SCARD_BDT_TAG_CHALLENGE_DATA macro 312

SCARD_BDT_TAG_CHALLENGE_TEMPLATE macro 313

SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA macro 313

SCARD_BDT_TAG_PROPRIETARY_BIOMETRIC_DATA_OBJECTS macro 313

SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA macro 313	SCARD_TAG_APPLICATION_RELATED_DATA macro 473
SCARD_BDT_TAG_STANDARD_BIOMETRIC_DATA_OBJECTS macro 313	SCARD_TAG_APPLICATION_TEMPLATE macro 473
SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_DATA macro 314	SCARD_TAG_APPLICATION_TRACK_1 macro 473
SCARD_BIT_BHT_TAG_BIOMETRIC_ALGORITHM_PARAMETERS_TEMPLATE macro 314	SCARD_TAG_APPLICATION_TRACK_2 macro 473
SCARD_BIT_BHT_TAG_BIOMETRIC_SUBTYPE macro 314	SCARD_TAG_APPLICATION_TRACK_3 macro 473
SCARD_BIT_BHT_TAG_BIOMETRIC_TYPE macro 314	SCARD_TAG_AUTHENTICATION_DATA macro 474
SCARD_BIT_BHT_TAG_CREATION_DATE macro 314	SCARD_TAG_BANK_IDENTIFIER_CODE macro 474
SCARD_BIT_BHT_TAG_CREATOR macro 315	SCARD_TAG_BIOMETRIC_DATA_TEMPLATE macro 474
SCARD_BIT_BHT_TAG_FORMAT_OWNER macro 315	SCARD_TAG_BIOMETRIC_INFORMATION_GROUP_TEMPLATE macro 474
SCARD_BIT_BHT_TAG_FORMAT_TYPE macro 315	SCARD_TAG_BIOMETRIC_INFORMATION_TEMPLATE macro 474
SCARD_BIT_BHT_TAG_INDEX macro 315	SCARD_TAG_CARD_CAPABILITIES macro 475
SCARD_BIT_BHT_TAG_PATRON_HEADER_VERSION macro 315	SCARD_TAG_CARD_DATA macro 475
SCARD_BIT_BHT_TAG_PRODUCT_IDENTIFIER macro 316	SCARD_TAG_CARD_EFFECTIVE_DATE macro 475
SCARD_BIT_BHT_TAG_PROPRIETARY_BHT macro 316	SCARD_TAG_CARD_EXPIRATION_DATE macro 475
SCARD_BIT_BHT_TAG_STANDARD_BHT macro 316	SCARD_TAG_CARD_ISSUERS_DATA macro 475
SCARD_BIT_BHT_TAG_VALIDITY_PERIOD macro 316	SCARD_TAG_CARD_SEQUENCE_NUMBER macro 475
SCARD_BIT_TAG_ALGORITHM_REFERENCE macro 316	SCARD_TAG_CARD_SERVICE_DATA macro 476
SCARD_BIT_TAG_BIOMETRIC_HEADER_TEMPLATE macro 317	SCARD_TAG_CARD_TRACK_1 macro 476
SCARD_BIT_TAG_REFERENCE_DATA_QUALIFIER macro 317	SCARD_TAG_CARD_TRACK_2 macro 476
SCARD_BIT_TAG_STANDARD_DATA_OBJECTS macro 317	SCARD_TAG_CARD_TRACK_3 macro 476
SCARD_BIT_TAG_TAG_ALLOCATION_AUTHORITY_DATA_OBJECTS macro 317	SCARD_TAG_CARDHOLDER_BIOMETRIC_DATA macro 476
SCARD_TAG_ACCOUNT_TYPE macro 471	SCARD_TAG_CARDHOLDER_CERTIFICATE macro 477
SCARD_TAG_ADDRESS macro 471	SCARD_TAG_CARDHOLDER_HANDWRITTEN_SIGNATURE_IMAGE macro 477
SCARD_TAG_ANSWER_TO_RESET macro 471	SCARD_TAG_CARDHOLDER_IMAGE_TEMPLATE macro 477
SCARD_TAG_APPLICATION_EFFECTIVE_DATE macro 471	SCARD_TAG_CARDHOLDER_NAME macro 477
SCARD_TAG_APPLICATION_EXPIRATION_DATE macro 472	SCARD_TAG_CARDHOLDER_NATIONALITY macro 477
SCARD_TAG_APPLICATION_FAMILY_IDENTIFIER macro 472	SCARD_TAG_CARDHOLDER_PORTRAIT_IMAGE macro 478
SCARD_TAG_APPLICATION_IDENTIFIER macro 472	SCARD_TAG_CARDHOLDER_PRIVATE_KEY macro 478
SCARD_TAG_APPLICATION_IMAGE macro 472	SCARD_TAG_CARDHOLDER_PRIVATE_KEY_TEMPLATE macro 478
SCARD_TAG_APPLICATION_IMAGE_TEMPLATE macro 472	SCARD_TAG_CARDHOLDER_PUBLIC_KEY macro 478
SCARD_TAG_APPLICATION_LABEL macro 473	SCARD_TAG_CARDHOLDER_PUBLIC_KEY_TEMPLATE macro 478
	SCARD_TAG_CARDHOLDER_RELATED_DATA macro 478
	SCARD_TAG_CARDHOLDER_REQUIREMENTS_EXCLUDED_FEATURES macro 479

SCARD_TAG_CARDHOLDER_REQUIREMENTS_INCLUDE_D_FEATURES macro 479	SCARD_TAG_HEADER_LIST macro 485
SCARD_TAG_CERTIFICATE_CONTENT macro 479	SCARD_TAG_HISTORICAL_BYTES macro 485
SCARD_TAG_CERTIFICATE_CONTENT_TEMPLATE macro 479	SCARD_TAG_INITIAL_ACCESS_DATA macro 485
SCARD_TAG_CERTIFICATE_HOLDER_AUTHORIZATION macro 479	SCARD_TAG_INTEGRATED_CIRCUIT_MANUFACTURER_I DENTIFIER macro 485
SCARD_TAG_COEXISTENT_TAG_ALLOCATION_AUTHORI TY macro 480	SCARD_TAG_INTERCHANGE_CONTROL macro 485
SCARD_TAG_COMMAND_TO_PERFORM macro 480	SCARD_TAG_INTERCHANGE_PROFILE macro 486
SCARD_TAG_COMPATIBLE_TAG_ALLOCATION_AUTHORI TY macro 480	SCARD_TAG_INTERINDUSTRY_DATA_OBJECTS macro 486
SCARD_TAG_COUNTRY_CODE macro 480	SCARD_TAG_INTERNATIONAL_BANK_ACCOUNT_NUMBE R macro 486
SCARD_TAG_COUNTRY_CODE_ALPHA2_FORMAT macro 480	SCARD_TAG_ISSUER_IDENTIFICATION_NUMBER macro 486
SCARD_TAG_COUNTRY_CODE_ALPHA3_FORMAT macro 481	SCARD_TAG_LANGUAGE_PREFERENCES macro 486
SCARD_TAG_COUNTRY_CODE_AND_NATIONAL_DATA macro 481	SCARD_TAG_LOGIN_DATA_PROPRIETARY macro 486
SCARD_TAG_CURRENCY_CODE macro 481	SCARD_TAG_LOGIN_TEMPLATE macro 487
SCARD_TAG_CURRENCY_EXPONENT macro 481	SCARD_TAG_MESSAGE_REFERENCE macro 487
SCARD_TAG_DATE_OF_BIRTH macro 481	SCARD_TAG_NAME macro 487
SCARD_TAG_DEPRECATED macro 482	SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_70 macro 487
SCARD_TAG_DIGITAL_SIGNATURE macro 482	SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_71 macro 487
SCARD_TAG_DIGITAL_SIGNATURE_BLOCK macro 482	SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_72 macro 488
SCARD_TAG_DISCRETIONARY_DATA macro 482	SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_74 macro 488
SCARD_TAG_DISCRETIONARY_DATA_OBJECTS macro 482	SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_75 macro 488
SCARD_TAG_DISPLAY_CONTROL macro 482	SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_76 macro 488
SCARD_TAG_DISPLAY_MESSAGE macro 483	SCARD_TAG_NON_INTERINDUSTRY_DATA_OBJECTS_77 macro 488
SCARD_TAG_DYNAMIC_AUTHENTICATION_TEMPLATE macro 483	SCARD_TAG_OFFSET_DATA_OBJECT macro 488
SCARD_TAG_DYNAMIC_EXTERNAL_AUTHENTICATION macro 483	SCARD_TAG_PIN_USAGE_POLICY macro 489
SCARD_TAG_DYNAMIC_INTERNAL_AUTHENTICATION macro 483	SCARD_TAG_PRE_ISSUING_DATA macro 489
SCARD_TAG_ELEMENT_LIST macro 484	SCARD_TAG_PRIMARY_ACCOUNT_NUMBER macro 489
SCARD_TAG_EXTENDED_HEADER_LIST macro 484	SCARD_TAG_PUBLIC_KEY_OF_CERTIFICATION_AUTHO RITY macro 489
SCARD_TAG_FCI_TEMPLATE macro 484	SCARD_TAG_QUALIFIED_NAME macro 489
SCARD_TAG_FCP_TEMPLATE macro 484	
SCARD_TAG_FILE_REFERENCE macro 484	
SCARD_TAG_FMD_TEMPLATE macro 484	

SCARD_TAG_SECURE_MESSAGING_TEMPLATE macro 490	BiometricAlgorithmParametersTemplate 1006
SCARD_TAG_SECURITY_ENVIRONMENT_TEMPLATE macro 490	BiometricSubtype 1006
SCARD_TAG_SECURITY_SUPPORT_TEMPLATE macro 490	BiometricType 1006
SCARD_TAG_SERVICE_CODE macro 490	CreationDate 1006
SCARD_TAG_SEX macro 490	Creator 1006
SCARD_TAG_SPECIAL_USER_REQUIREMENTS macro 490	FormatOwner 1006
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION macro 491	FormatType 1007
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_FIR ST_ASSOCIATED_DATA macro 491	Index 1007
SCARD_TAG_STATIC_INTERNAL_AUTHENTICATION_SEC OND_ASSOCIATED_DATA macro 491	PatronHeaderVersion 1007
SCARD_TAG_STATUS_INFORMATION macro 491	ProductIdentifier 1007
SCARD_TAG_TAG_LIST macro 492	ProprietaryBht 1007
SCARD_TAG_TIMER macro 492	StandardBht 1007
SCARD_TAG_TRANSACTION_COUNTER macro 492	ValidityPeriod 1007
SCARD_TAG_TRANSACTION_DATE macro 492	SCardBitTags class 1007
SCARD_TAG_UNIFORM_RESOURCE_LOCATOR macro 492	about SCardBitTags class 1007
SCARD_TAG_WRAPPER macro 493	AlgorithmReference 1008
SCardBdtCTTags class 1003	BiometricHeaderTemplate 1008
about SCardBdtCTTags class 1003	ReferenceDataQualifier 1008
Challenge 1003	StandardDataObjects 1008
ChallengeQualifier 1003	TagAllocationAuthorityDataObjects 1008
SCardBdtTags class 1003	SCardChallengeQualifier enumeration 312
about SCardBdtTags class 1003	SCardChallengeQualifier_ enumeration 312
ChallengeData 1004	SCardTags class 972
ChallengeTemplate 1004	about SCardTags class 972
ProprietaryBiometricData 1004	AccountType 977
ProprietaryBiometricDataObjects 1004	Address 977
StandardBiometricData 1004	AnswerToReset 977
StandardBiometricDataObjects 1005	ApplicationEffectiveDate 977
SCardBiometricSubtype enumeration 310	ApplicationExpirationDate 978
SCardBiometricSubtype_ enumeration 310	ApplicationFamilyIdentifier 978
SCardBiometricType enumeration 311	ApplicationIdentifier 978
SCardBiometricType_ enumeration 311	ApplicationImage 978
SCardBitBhtTags class 1005	ApplicationImageTemplate 978
about SCardBitBhtTags class 1005	ApplicationLabel 978
BiometricAlgorithmParametersData 1006	ApplicationRelatedData 978
	ApplicationTemplate 978
	ApplicationTrack1 978
	ApplicationTrack2 979
	ApplicationTrack3 979
	AuthenticationData 979
	BankIdentifierCode 979

BiometricDataTemplate 979	DigitalSignature 984
BiometricInformationGroupTemplate 979	DigitalSignatureBlock 984
BiometricInformationTemplate 979	DiscretionaryData 984
CardCapabilities 979	DiscretionaryDataObjects 984
CardData 980	DisplayControl 984
CardEffectiveDate 980	DisplayMessage 984
CardExpirationDate 980	DynamicAuthenticationTemplate 984
CardholderBiometricData 980	DynamicExternalAuthentication 985
CardholderCertificate 980	DynamicInternalAuthentication 985
CardholderHandwrittenSignatureImage 980	DynamicMutualAuthentication 985
CardholderImageTemplate 980	ElementList 985
CardholderName 980	ExtendedHeaderList 985
CardholderNationality 980	FciTemplate 985
CardholderPortraitImage 981	FcpTemplate 985
CardholderPrivateKey 981	FileReference 985
CardholderPrivateKeyTemplate 981	FmdTemplate 985
CardholderPublicKey 981	HeaderList 986
CardholderPublicKeyTemplate 981	HistoricalBytes 986
CardholderRelatedData 981	InitialAccessData 986
CardholderRequirementsExcludedFeatures 981	IntegratedCircuitManufacturerIdentifier 986
CardholderRequirementsIncludedFeatures 981	InterchangeControl 986
CardIssuersData 981	InterchangeProfile 986
CardSequenceNumber 982	InterindustryDataObjects 986
CardServiceData 982	InternationalBankAccountNumber 986
CardTrack1 982	IssuerIdentificationNumber 986
CardTrack2 982	LanguagePreferences 987
CardTrack3 982	LoginDataProprietary 987
CertificateContent 982	LoginTemplate 987
CertificateContentTemplate 982	MessageReference 987
CertificateHolderAuthorization 982	Name 987
CoexistentTagAllocationAuthority 983	NonInterindustryDataObjects70 987
CommandToPerform 983	NonInterindustryDataObjects71 987
CompatibleTagAllocationAuthority 983	NonInterindustryDataObjects72 987
CountryCode 983	NonInterindustryDataObjects74 988
CountryCodeAlpha2Format 983	NonInterindustryDataObjects75 988
CountryCodeAlpha3Format 983	NonInterindustryDataObjects76 988
CountryCodeAndNationalData 983	NonInterindustryDataObjects77 988
CurrencyCode 983	OffsetDataObject 988
CurrencyExponent 983	PinUsagePolicy 988
DateOfBirth 984	PreIssuingData 988
Deprecated 984	PrimaryAccountNumber 988

PublicKeyOfCertificationAuthority 988	scbtNone enumeration member 311
QualifiedName 989	scbtPalmPrint enumeration member 311
SecureMessagingTemplate 989	scbtRetina enumeration member 311
SecurityEnvironmentTemplate 989	scbtSignatureDynamics enumeration member 311
SecuritySupportTemplate 989	scbtThermalFacelImage enumeration member 311
ServiceCode 989	scbtThermalHandImage enumeration member 311
Sex 989	scbtVeinPattern enumeration member 311
SpecialUserRequirements 989	scbtVoice enumeration member 311
StaticInternalAuthentication 989	sccqUnspecified enumeration member 312
StaticInternalAuthenticationFirstAssociatedData 989	sccqUtf8Coding enumeration member 312
StaticInternalAuthenticationSecondAssociatedData 990	SignatureDynamics enumeration member 1009
StatusInformation 990	SimpleTag structure 991
TagList 990	!= 992
Timer 990	== 993
TransactionCounter 990	about SimpleTag structure 991
TransactionDate 990	byte 993
UniformResourceLocator 990	Equals 992
Wrapper 990	GetHashCode 992
SCardTransmitApu function 380	IsValid 993
Scarf enumeration member 805	SimpleTag 991
scbsLeft enumeration member 310	ToString 992
scbsLittleFinger enumeration member 310	SimpleTagIsValid function 505
scbsMiddleFinger enumeration member 310	SimpleTlv class 993
scbsNone enumeration member 310	about SimpleTlv class 993
scbsPointerFinger enumeration member 310	Clone 996
scbsRight enumeration member 310	FromHandle 996
scbsRingFinger enumeration member 310	GetSizeMany 996
scbsThumb enumeration member 310	GetValue 997
scbtBodyOdor enumeration member 311	Length 1000
scbtDna enumeration member 311	LoadMany 997, 998
scbtEarShape enumeration member 311	Save 998
scbtFacialFeatures enumeration member 311	SaveMany 999
scbtFingerGeometry enumeration member 311	SetLength 999
scbtFingerprint enumeration member 311	SetValue 1000
scbtFootPrint enumeration member 311	SimpleTlv 995
scbtGait enumeration member 311	Size 1000
scbtHandGeometry enumeration member 311	Tag 1000
scbtIris enumeration member 311	UseThreeByteLengthAlways 1001
scbtKeystrokeDynamics enumeration member 311	ValuePtr 1001
scbtLipMovement enumeration member 311	SimpleTlv Module 504
scbtMultipleBiometrics enumeration member 311	SimpleTlv.h 513

SimpleTlvClone function 505
 SimpleTlvCreate function 506
 SimpleTlvCreateFromMemory function 506
 SimpleTlvGetLength function 507
 SimpleTlvGetSize function 507
 SimpleTlvGetSizeMany function 508
 SimpleTlvGetTag function 508
 SimpleTlvGetUseThreeByteLengthAlways function 509
 SimpleTlvGetValue function 509
 SimpleTlvGetValuePtr function 510
 SimpleTlvLoadManyFromMemory function 510
 SimpleTlvSaveManyToMemory function 511
 SimpleTlvSaveToMemory function 511
 SimpleTlvSetLength function 512
 SimpleTlvSetUseThreeByteLengthAlways function 512
 SimpleTlvSetValue function 513
 Single computer license 3
 SiS enumeration member 638
 System Requirements 1
 Skeletonized enumeration member 747
 Small enumeration member 747, 753
 Smartcard CAP Loader 12
 Smartcard Faces Client 11
 Smartcard Fingers Client 11
 Smile enumeration member 805
 SmileOpenedJaw enumeration member 805
 Specified enumeration member 805
 Squinting enumeration member 805
 Strawberry enumeration member 811
 Support 1032
 Supported Platforms 1
 Swipe enumeration member 747

T

TeethVisible enumeration member 805
 TemplateCreated enumeration member 746, 752
 TentedArch enumeration member 749, 806
 ThermalFacelImage enumeration member 1009
 ThermalHandImage enumeration member 1009
 Thumb enumeration member 1009
 Tiff class 883

about Tiff class 883
 LoadImage 883, 884
 Tiff Module 242
 Tiff.h 244
 TiffLoadImageFromFile function 242
 TiffLoadImageFromMemory function 243
 TiffLoadImageFromStream function 243
 TooFewMinutiae enumeration member 746
 TopDown enumeration member 884
 Transmeta enumeration member 638
 Tutorials 10

U

UlnarLoop enumeration member 749, 806
 Umc enumeration member 638
 Undefined enumeration member 812
 Universal enumeration member 1001
 Unknown enumeration member 638, 749, 750, 804, 806, 807, 808, 809, 810, 811
 UnknownPalm enumeration member 750, 809
 UnknownPhoto enumeration member 811
 UnknownVideo enumeration member 811
 Unspecified enumeration member 752, 804, 805, 808, 810, 811, 1010
 Using 6
 Using On-Card Bio API Library 7
 Using On-card Bio API Sample Applets 7
 Utf8Coding enumeration member 1010

V

VeinPattern enumeration member 1009
 Vendor enumeration member 811
 Via enumeration member 638
 Voice enumeration member 1009
 void * pParam) variable 150

W

White enumeration member 811
 Whorl enumeration member 749, 806

X

XOrYCoordinateExtension enumeration member 814