MegaMatcher 1.1 SDK

MegaMatcher 1.1 SDK

Published May 12, 2006. Version 1.1.0.0
Copyright © 2005-2006 Neurotechnologija

Table of Contents

1. About 1
    1.1. Introduction 1
        1.1.1. Large-scale biometric systems in the world 1
        1.1.2. LSBS Requirements 1
        1.1.3. MegaMatcher Features 2
    1.2. Platforms Supported 3
    1.3. System Requirements 3
    1.4. Licensing 4
2. What's New 5
3. Overview 6
    3.1. Server Components 6
    3.2. Client Components 7
        3.2.1. Fingerprint Features Extraction 7
        3.2.2. Template Management 7
        3.2.3. Fingerprint Scanner Support 10
            3.2.3.1. Scanners API for Linux OS 11
        3.2.4. Fingerprint View Control 13
    3.3. Additional Components 13
        3.3.1. Image Support 14
            3.3.1.1. Image 14
            3.3.1.2. Image Format 15
            3.3.1.3. Image File 16
            3.3.1.4. Low-Level Image Input-Output 17
        3.3.2. Template Conversion 17
    3.4. Samples 17
4. Using 19
    4.1. LAN License Manager 19
        4.1.1. LAN License Manager Monitor 19
            4.1.1.1. Main window 20
            4.1.1.2. Configuration window 20
    4.2. Server Components 21
        4.2.1. Installing 21
        4.2.2. Configuring 23
            4.2.2.1. Server 23
            4.2.2.2. Node 26
        4.2.3. Running 30
        4.2.4. Client-Server Communication 31
            4.2.4.1. Sending task and getting results 31
            4.2.4.2. Inserting template 32
            4.2.4.3. Getting information 33
        4.2.5. Packets 33
            4.2.5.1. packet_type Enumeration 33
            4.2.5.2. PACKET_ADMIN_CMD 34
                4.2.5.2.1. adm_type Enumeration 34
                4.2.5.2.2. adm_info_type Enumeration 35

Copyright © 2006 Neurotechnologija

4.2.5.3. PACKET_ADMIN_RESULT_INFO 35
4.2.5.3.1. admin_info Structure 36
4.2.5.3.2. node_inf Structure 36
4.2.5.3.3. c_node_state Structure 36
4.2.5.3.4. c_info_r Structure 37
4.2.5.3.5. c_info_inc Structure 37
4.2.5.4. PACKET_DB_CHANGED 38
4.2.5.4.1. db_changed Structure 38
4.2.5.4.2. template_id Structure 38
4.2.5.5. PACKET_RESULT 39
4.2.5.5.1. result Structure 39
4.2.5.6. PACKET_RESULT_DELETE 39
4.2.5.6.1. result_del Structure 39
4.2.5.7. PACKET_RESULT_HEADER 40
4.2.5.7.1. result_header Structure 40
4.2.5.8. PACKET_RESULT_REQ 41
4.2.5.8.1. result_req Structure 41
4.2.5.9. PACKET_TASK 41
4.2.5.9.1. task Structure 42
4.2.5.9.2. task_mode Enumeration 42
4.2.5.10. PACKET_TASK_ID 43
4.2.5.10.1. task_id Structure 43
4.2.5.11. PACKET_TASK_PROGRESS 43
4.2.5.12. PACKET_TASK_PROGRESS_REQ 43
4.3. Databases 44
4.3.1. MySQL 45
4.3.1.1. Installing 45
4.3.1.2. Configuring 45
4.3.1.2.1. Server Configuration 45
4.3.1.2.2. Node(-s) Configuration 46
4.3.1.3. Compiling MySQL DB drivers 47
4.3.1.3.1. Compiling drivers on Windows 47
4.3.1.3.2. Compiling server MySQL DB driver on Linux 47
4.3.1.3.3. Compiling and linking node with MySQL DB module on Linux 47
4.3.2. Oracle 48
4.3.2.1. Installing 48
4.3.2.2. Server Configuration 48
4.3.2.3. Compiling Oracle DB driver 48
4.3.3. SQLite 49
4.3.3.1. Installing 49
4.3.3.2. Node(-s) configuration 49
4.3.3.3. Compiling SQLite DB driver 49
4.3.4. Other providers 50
4.3.4.1. Server DB Driver 50
4.3.4.1.1. connect function 50
4.3.4.1.2. close function 51
4.3.4.1.3. field_count function 51
4.3.4.1.4. fields function 51
4.3.4.1.5. fetch function 52

Copyright © 2006 Neurotechnologija

4.3.4.1.6. query function 53
4.3.4.1.7. query_close function 53
4.3.4.1.8. get_object function 53
4.3.4.1.9. on_delete function 54
4.3.4.2. Node DB module 54
4.3.4.2.1. Functions to work with node 54
4.3.4.2.2. Node DB module functions to be implemented 57
4.4. Client Components 60
4.4.1. Template Management 60
4.4.1.1. Packing NFRecord to NTemplate 60
4.4.1.2. Retrieving Packed NFRecord from Packed NTemplate 63
4.4.1.3. Converting Packed NFRecord from Version 1.0 Format to Current (Version 2.0) Format 65
4.4.1.4. Packing Version 1.0 Format NFRecord to NTemplate 66
4.4.1.5. Converting Packed NFRecord from Current (Version 2.0) Format to Version 1.0 Format 66
4.4.1.6. Retrieving Packed NFRecord in Version 1.0 Format from Packed NTemplate 67
4.5. Additional Components 67
4.5.1. Template Conversion 67
4.5.1.1. Converting NFRecord to FMRecord 68
4.5.1.2. Converting Version 1.0 Format NFRecord to FMRecord 70
4.5.1.3. Converting Two Version 1.0 Format NFRecords to FMRecord 70
4.5.1.4. Converting NTemplate to FMRecord 72
4.5.1.5. Converting FMRecord to NFRecord 73
4.5.1.6. Converting Two-Finger FMRecord to NFRecords in Version 1.0 Format 75
4.5.1.7. Converting FMRecord to NTemplate 77
4.5.1.8. Converting NFRecord to ANTemplate 78
4.5.1.9. Converting Version 1.0 Format NFRecord to ANTemplate 79
4.5.1.10. Converting NTemplate to ANTemplate 79
4.5.1.11. Converting ANTemplate to NFRecord 81
4.5.1.12. Converting ANTemplate to NTemplate 82
4.6. Translation of User Interface 83
5. Samples 85
5.1. Oracle and MySQL database tool for Linux OS 85
5.1.1. Files for Linux OS 86
5.2. Sample of Cluster Administrating tool 86
5.3. Client sample for Linux OS 87
5.3.1. Using Command Line Interface for controlling the Client application 88
5.3.1.1. Client command line options 88
5.3.1.2. Submitting the tasks 89
5.3.1.3. Retrieving the results 89
5.3.2. Files 89
5.4. Client application for .NET 89
5.4.1. Description 89
5.4.2. Running the demo application 90
5.4.3. User interface 90
5.4.3.1. Main window 90
5.4.3.2. Main menu 91

5.4.3.3. Buttons 92
5.4.3.4. Features 93
5.4.3.5. Code description 93
5.4.4. Cluster communication 94
5.4.4.1. MatchingClusterClient Class 94
5.4.4.1.1. MatchingClusterClient Constructors 95
5.4.4.1.2. ServerAddress Property 95
5.4.4.1.3. ServerPort Property 95
5.4.4.1.4. Tasks Property 96
5.4.4.1.5. Dispose Method 96
5.4.4.1.6. HasPendingTask Method 96
5.4.4.2. MatchingTask Class 96
5.4.4.2.1. MatchingTask Constructors 97
5.4.4.2.2. Error Property 97
5.4.4.2.3. IsComplete Property 98
5.4.4.2.4. IsStarted Property 98
5.4.4.2.5. Name Property 98
5.4.4.2.6. Parameters Property 98
5.4.4.2.7. Progress Property 98
5.4.4.2.8. Query Property 99
5.4.4.2.9. Results Property 99
5.4.4.2.10. SearchUntilFirstMatch Property 99
5.4.4.2.11. Template Property 99
5.4.4.3. MatchingTaskParameter Struct 99
5.4.4.3.1. MatchingClusterClient Constructors 100
5.4.4.3.2. ID Property 100
5.4.4.3.3. PartID Property 100
5.4.4.3.4. Value Property 101
5.4.5. Fingerprint record editor 101
5.4.5.1. Buttons 102
5.5. Client application for Visual Basic 6 102
5.5.1. Description 102
5.5.2. Running the demo application 102
5.5.3. User interface 104
5.5.4. Wrappers 104
5.5.4.1. modNFTemplate Module 104
5.5.4.1.1. NFTemplatePack Method 104
5.5.4.2. modNImages Module 104
5.5.4.2.1. BmpSaveImageToHBitmap Method 105
5.5.4.2.2. NImageCreateFromData Method 105
5.5.4.2.3. NImageCreateFromFile Method 106
5.5.4.2.4. NImageCreateFromImage Method 107
5.5.4.2.5. NImageSaveToFile Method 107
5.5.4.2.6. NImagesIsRegistered Method 108
5.5.4.2.7. NImagesRegister Method 108
5.5.4.2.8. NImagesUnregister Method 109
5.5.4.2.9. Picture Method 109
5.5.4.3. modNTemplate Module 109
5.5.4.3.1. NTemplatePack Method 110
5.5.4.4. modWSQ Module 110

Copyright © 2006 Neurotechnologija

5.5.4.4.1. WsqLoadImageFromFile Method 110
5.5.4.4.2. WsqSaveImageToFile Method 111
5.5.4.5. NFExractor Class 111
5.5.4.5.1. NFExtractor Constructor 112
5.5.4.5.2. CopyParameters Method 112
5.5.4.5.3. ErrorOccured Event 113
5.5.4.5.4. Extract Method 113
5.5.4.5.5. ExtractFromImage Method 114
5.5.4.5.6. GetMaxTemplateSize Property 115
5.5.4.5.7. GetParameter Property 115
5.5.4.5.8. Handle Property 115
5.5.4.5.9. IsRegistered Property 115
5.5.4.5.10. Register Method 116
5.5.4.5.11. Reset Method 116
5.5.4.5.12. SetParameter Method 116
5.5.4.5.13. Unregister Method 117
5.5.4.6. NFRecord Class 117
5.5.4.6.1. NFRecord Constructor 118
5.5.4.6.2. ErrorOccured Event 118
5.5.4.6.3. RecordCreateFromMemory Method 118
5.5.4.6.4. RecordGetMinutia Method 119
5.5.4.6.5. RecordGetMinutiaCount Property 119
5.5.4.6.6. RecordGetMinutiae Method 119
5.5.4.7. VFExractor Class 120
6. Reference (C/C++) 121
6.1. ANTemplate Library 122
6.1.1. ANTemplate Module 123
6.1.1.1. ANTemplateCreateFromFile Function 123
6.1.1.2. ANTemplateCreateFromNFRecord Function 124
6.1.1.3. ANTemplateCreateFromNFTemplate Function 125
6.1.1.4. ANTemplateCreateFromNTemplate Function 127
6.1.1.5. ANTemplateFree Function 128
6.1.1.6. ANTemplateSaveToFile Function 128
6.1.1.7. ANTemplateToNFTemplate Function 129
6.1.1.8. ANTemplateToNTemplate Function 130
6.2. FMRecord Library 130
6.2.1. FMRecord Module 131
6.2.1.1. FMRecordAddFingerView Function 134
6.2.1.2. FMRecordAddFingerViewCopy Function 134
6.2.1.3. FMRecordCalculateSize Function 135
6.2.1.4. FMRecordClearFingerViews Function 136
6.2.1.5. FMRecordClone Function 137
6.2.1.6. FMRecordCreate Function 138
6.2.1.7. FMRecordCreateFromMemory Function 139
6.2.1.8. FMRecordCreateFromNFRecord Function 140
6.2.1.9. FMRecordCreateFromNFTemplate Function 141
6.2.1.10. FMRecordFree Function 142
6.2.1.11. FMRecordGetCaptureEquipmentCompliance Function 143
6.2.1.12. FMRecordGetCaptureEquipmentId Function 143
6.2.1.13. FMRecordGetCbeffProductId Function 144

6.2.1.14. FMRecordGetFingerView Function 145
6.2.1.15. FMRecordGetFingerViewCapacity Function 145
6.2.1.16. FMRecordGetFingerViewCount Function 146
6.2.1.17. FMRecordGetResolutionX Function 147
6.2.1.18. FMRecordGetResolutionY Function 148
6.2.1.19. FMRecordGetSize Function 149
6.2.1.20. FMRecordGetSizeX Function 149
6.2.1.21. FMRecordGetSizeY Function 150
6.2.1.22. FMRecordRemoveFingerView Function 151
6.2.1.23. FMRecordSaveToMemory Function 151
6.2.1.24. FMRecordSetCaptureEquipmentCompliance Function 153
6.2.1.25. FMRecordSetCaptureEquipmentId Function 154
6.2.1.26. FMRecordSetCbeffProductId Function 154
6.2.1.27. FMRecordSetFingerViewCapacity Function 155
6.2.1.28. FMRecordToNFTemplate Function 156
6.2.1.29. FMRecordToNTemplate Function 157
6.2.2. FmrFingerView Module 158
6.2.2.1. FmrCore Structure 162
6.2.2.1.1. FmrCore.Angle Field 163
6.2.2.1.2. FmrCore.X Field 163
6.2.2.1.3. FmrCore.Y Field 163
6.2.2.2. FmrDelta Structure 163
6.2.2.2.1. FmrDelta.Angle1 Field 164
6.2.2.2.2. FmrDelta.Angle2 Field 164
6.2.2.2.3. FmrDelta.Angle3 Field 165
6.2.2.2.4. FmrDelta.X Field 165
6.2.2.2.5. FmrDelta.Y Field 165
6.2.2.3. FmrFingerViewAddCore Function 166
6.2.2.4. FmrFingerViewAddDelta Function 166
6.2.2.5. FmrFingerViewAddMinutia Function 167
6.2.2.6. FmrFingerViewClearCores Function 168
6.2.2.7. FmrFingerViewClearDeltas Function 168
6.2.2.8. FmrFingerViewClearMinutiae Function 169
6.2.2.9. FmrFingerViewGetCore Function 170
6.2.2.10. FmrFingerViewGetCoreCapacity Function 171
6.2.2.11. FmrFingerViewGetCoreCount Function 171
6.2.2.12. FmrFingerViewGetCores Function 172
6.2.2.13. FmrFingerViewGetDelta Function 173
6.2.2.14. FmrFingerViewGetDeltaCapacity Function 174
6.2.2.15. FmrFingerViewGetDeltaCount Function 175
6.2.2.16. FmrFingerViewGetDeltas Function 176
6.2.2.17. FmrFingerViewGetFingerPosition Function 176
6.2.2.18. FmrFingerViewGetFingerQuality Function 177
6.2.2.19. FmrFingerViewGetImpressionType Function 178
6.2.2.20. FmrFingerViewGetMaxSize Function 179
6.2.2.21. FmrFingerViewGetMinutia Function 180
6.2.2.22. FmrFingerViewGetMinutiaCapacity Function 181
6.2.2.23. FmrFingerViewGetMinutiaCount Function 182
6.2.2.24. FmrFingerViewGetMinutiaEightNeighbour Function 183
6.2.2.25. FmrFingerViewGetMinutiaEightNeighbours Function 184

6.2.2.26. FmrFingerViewGetMinutiaFourNeighbour Function 185
6.2.2.27. FmrFingerViewGetMinutiaFourNeighbours Function 186
6.2.2.28. FmrFingerViewGetMinutiae Function 187
6.2.2.29. FmrFingerViewGetViewNumber Function 187
6.2.2.30. FmrFingerViewHasEightNeighbourRidgeCounts Function .188
6.2.2.31. FmrFingerViewHasFourNeighbourRidgeCounts Function ..189
6.2.2.32. FmrFingerViewInsertCore Function 190
6.2.2.33. FmrFingerViewInsertDelta Function 190
6.2.2.34. FmrFingerViewInsertMinutia Function 191
6.2.2.35. FmrFingerViewRemoveCore Function 192
6.2.2.36. FmrFingerViewRemoveDelta Function 193
6.2.2.37. FmrFingerViewRemoveMinutia Function 194
6.2.2.38. FmrFingerViewSetCore Function 195
6.2.2.39. FmrFingerViewSetCoreCapacity Function 195
6.2.2.40. FmrFingerViewSetDelta Function 196
6.2.2.41. FmrFingerViewSetDeltaCapacity Function 197
6.2.2.42. FmrFingerViewSetFingerPosition Function 198
6.2.2.43. FmrFingerViewSetFingerQuality Function 199
6.2.2.44. FmrFingerViewSetHasEightNeighbourRidgeCounts Function 199
6.2.2.45. FmrFingerViewSetHasFourNeighbourRidgeCounts Function 200
6.2.2.46. FmrFingerViewSetImpressionType Function 201
6.2.2.47. FmrFingerViewSetMinutia Function 202
6.2.2.48. FmrFingerViewSetMinutiaCapacity Function 202
6.2.2.49. FmrFingerViewSetMinutiaEightNeighbour Function 203
6.2.2.50. FmrFingerViewSetMinutiaFourNeighbour Function 204
6.2.2.51. FmrFingerViewSetViewNumber Function 206
6.2.2.52. FmrFingerViewToNFRecord Function 206
6.2.2.53. FmrMinutia Structure 207
6.2.2.53.1. FmrMinutia.Angle Field 208
6.2.2.53.2. FmrMinutia.Quality Field 208
6.2.2.53.3. FmrMinutia.Type Field 208
6.2.2.53.4. FmrMinutia.X Field 208
6.2.2.53.5. FmrMinutia.Y Field 209
6.3. NCore Library 209
6.3.1. NCore Module 210
6.3.2. NErrors Module 211
6.3.3. NParameters Module 212
6.3.3.1. NParameterMakeId Macro 213
6.3.4. NTypes Module 213
6.3.4.1. NByteOrder Enumeration 218
6.3.4.2. NFileAccess Enumeration 218
6.3.4.3. NIndexPair Structure 218
6.3.4.3.1. NIndexPair.Index1 Field 219
6.3.4.3.2. NIndexPair.Index2 Field 219
6.3.4.4. NRational Structure 219
6.3.4.4.1. NRational.Denominator Field 220
6.3.4.4.2. NRational.Numerator Field 220
6.3.4.5. NURational Structure 220
6.3.4.5.1. NURational.Denominator Field 220

Copyright © 2006 Neurotechnologija

6.3.4.5.2. NURational.Numerator Field 221
6.4. NFExtractor Library 221
6.4.1. NFExtractor Module 221
6.4.1.1. NfeCopyParameters Function 224
6.4.1.2. NfeCreate Function 225
6.4.1.3. NfeExtract Function 226
6.4.1.4. NfeExtractFromImage Function 228
6.4.1.5. NfeExtractUnpacked Function 229
6.4.1.6. NfeExtractUnpackedFromImage Function 231
6.4.1.7. NfeFree Function 232
6.4.1.8. NfeGetMaxTemplateSize Function 233
6.4.1.9. NfeGetParameter Function 233
6.4.1.10. NfeIsRegistered Function 235
6.4.1.11. NfeRegister Function 235
6.4.1.12. NfeReset Function 236
6.4.1.13. NfeReturnedImage Enumeration 237
6.4.1.14. NfeSetParameter Function 237
6.4.1.15. NfeUnregister Function 238
6.4.1.16. NfeUpdateTemplate Function 239
6.4.1.17. NfeUpdateTemplateFromImage Function 240
6.4.1.18. NfeUpdateUnpackedTemplate Function 241
6.4.1.19. NfeUpdateUnpackedTemplateFromImage Function 242
6.5. NFMatcher Library 243
6.5.1. NFMatcher Module 244
6.5.1.1. NfmCopyParameters Function 247
6.5.1.2. NfmCreate Function 248
6.5.1.3. NfmFree Function 248
6.5.1.4. NfmGetParameter Function 249
6.5.1.5. NfmIdentifyEnd Function 250
6.5.1.6. NfmIdentifyNext Function 251
6.5.1.7. NfmIdentifyStart Function 252
6.5.1.8. NfmIsRegistered Function 253
6.5.1.9. NfmMatchDetails Structure 254
6.5.1.9.1. NfmMatchDetails.MatedMinutiaCount Field 254
6.5.1.9.2. NfmMatchDetails.MatedMinutiae Field 254
6.5.1.9.3. NfmMatchDetailsScore.Score Field 254
6.5.1.10. NfmMatchDetailsFree Function 255
6.5.1.11. NfmRegister Function 255
6.5.1.12. NfmReset Function 256
6.5.1.13. NfmSetParameter Function 257
6.5.1.14. NfmSpeed Enumeration 258
6.5.1.15. NfmUnregister Function 258
6.5.1.16. NfmVerify Function 259
6.6. NFRecord Library 260
6.6.1. NFRecord Module 261
6.6.1.1. NFCore Structure 267
6.6.1.1.1. NFCore.Angle Field 268
6.6.1.1.2. NFCore.X Field 268
6.6.1.1.3. NFCore.Y Field 268
6.6.1.2. NFDelta Structure 269

6.6.1.2.1. NFDelta.Angle1 Field 269
6.6.1.2.2. NFDelta.Angle2 Field 270
6.6.1.2.3. NFDelta.Angle3 Field 270
6.6.1.2.4. NFDelta.X Field 270
6.6.1.2.5. NFDelta.Y Field 271
6.6.1.3. NFDoubleCore Structure 271
6.6.1.3.1. NFDoubleCore.X Field 271
6.6.1.3.2. NFDoubleCore.Y Field 272
6.6.1.4. NFImpressionType Enumeration 272
6.6.1.5. NFMinutia Structure 273
6.6.1.5.1. NFMinutia.Angle Field 273
6.6.1.5.2. NFMinutia.Curvature Field 274
6.6.1.5.3. NFMinutia.G Field 274
6.6.1.5.4. NFMinutia.Quality Field 274
6.6.1.5.5. NFMinutia.Type Field 275
6.6.1.5.6. NFMinutia.X Field 275
6.6.1.5.7. NFMinutia.Y Field 275
6.6.1.6. NFMinutiaFormat Enumeration 276
6.6.1.7. NFMinutiaNeighbour Structure 276
6.6.1.7.1. NFMinutiaNeighbour.Index Field 277
6.6.1.7.2. NFMinutiaNeighbour.RidgeCount Field 277
6.6.1.8. NFMinutiaType Enumeration 277
6.6.1.9. NFPatternClass Enumeration 277
6.6.1.10. NFPosition Enumeration 278
6.6.1.11. NFRecordAddCore Function 279
6.6.1.12. NFRecordAddDelta Function 280
6.6.1.13. NFRecordAddDoubleCore Function 281
6.6.1.14. NFRecordAddMinutia Function 281
6.6.1.15. NFRecordCheck Function 282
6.6.1.16. NFRecordClearCores Function 283
6.6.1.17. NFRecordClearDeltas Function 284
6.6.1.18. NFRecordClearDoubleCores Function 284
6.6.1.19. NFRecordClearMinutiae Function 285
6.6.1.20. NFRecordClone Function 285
6.6.1.21. NFRecordCreate Function 286
6.6.1.22. NFRecordCreateFromMemory Function 288
6.6.1.23. NFRecordFree Function 289
6.6.1.24. NFRecordGetCore Function 289
6.6.1.25. NFRecordGetCoreCapacity Function 290
6.6.1.26. NFRecordGetCoreCount Function 291
6.6.1.27. NFRecordGetCores Function 292
6.6.1.28. NFRecordGetDelta Function 293
6.6.1.29. NFRecordGetDeltaCapacity Function 294
6.6.1.30. NFRecordGetDeltaCount Function 294
6.6.1.31. NFRecordGetDeltas Function 295
6.6.1.32. NFRecordGetDoubleCore Function 296
6.6.1.33. NFRecordGetDoubleCoreCapacity Function 297
6.6.1.34. NFRecordGetDoubleCoreCount Function 298
6.6.1.35. NFRecordGetDoubleCores Function 299
6.6.1.36. NFRecordGetG Function 299

Copyright © 2006 Neurotechnologija

6.6.1.37. NFRecordGetGMem Function 300
6.6.1.38. NFRecordGetHeight Function 301
6.6.1.39. NFRecordGetHeightMem Function 302
6.6.1.40. NFRecordGetHorzResolution Function 303
6.6.1.41. NFRecordGetHorzResolutionMem Function 303
6.6.1.42. NFRecordGetImpressionType Function 304
6.6.1.43. NFRecordGetImpressionTypeMem Function 305
6.6.1.44. NFRecordGetMaxSize Function 306
6.6.1.45. NFRecordGetMaxSizeV1 Function 308
6.6.1.46. NFRecordGetMinutia Function 310
6.6.1.47. NFRecordGetMinutiaCapacity Function 311
6.6.1.48. NFRecordGetMinutiaCount Function 311
6.6.1.49. NFRecordGetMinutiaFormat Function 312
6.6.1.50. NFRecordGetMinutiaNeighbour Function 313
6.6.1.51. NFRecordGetMinutiaNeighbourCount Function 314
6.6.1.52. NFRecordGetMinutiaNeighbours Function 315
6.6.1.53. NFRecordGetMinutiae Function 316
6.6.1.54. NFRecordGetPatternClass Function 316
6.6.1.55. NFRecordGetPatternClassMem Function 317
6.6.1.56. NFRecordGetPosition Function 318
6.6.1.57. NFRecordGetPositionMem Function 319
6.6.1.58. NFRecordGetQuality Function 320
6.6.1.59. NFRecordGetQualityMem Function 320
6.6.1.60. NFRecordGetRidgeCountsType Function 321
6.6.1.61. NFRecordGetSize Function 322
6.6.1.62. NFRecordGetSizeV1 Function 323
6.6.1.63. NFRecordGetVertResolution Function 324
6.6.1.64. NFRecordGetVertResolutionMem Function 324
6.6.1.65. NFRecordGetWidth Function 325
6.6.1.66. NFRecordGetWidthMem Function 326
6.6.1.67. NFRecordInsertCore Function 327
6.6.1.68. NFRecordInsertDelta Function 328
6.6.1.69. NFRecordInsertDoubleCore Function 329
6.6.1.70. NFRecordInsertMinutia Function 330
6.6.1.71. NFRecordRemoveCore Function 331
6.6.1.72. NFRecordRemoveDelta Function 331
6.6.1.73. NFRecordRemoveDoubleCore Function 332
6.6.1.74. NFRecordRemoveMinutia Function 333
6.6.1.75. NFRecordSaveToMemory Function 334
6.6.1.76. NFRecordSaveToMemoryV1 Function 335
6.6.1.77. NFRecordSetCore Function 337
6.6.1.78. NFRecordSetCoreCapacity Function 337
6.6.1.79. NFRecordSetDelta Function 338
6.6.1.80. NFRecordSetDeltaCapacity Function 339
6.6.1.81. NFRecordSetDoubleCore Function 340
6.6.1.82. NFRecordSetDoubleCoreCapacity Function 341
6.6.1.83. NFRecordSetG Function 342
6.6.1.84. NFRecordSetImpressionType Function 343
6.6.1.85. NFRecordSetMinutia Function 343
6.6.1.86. NFRecordSetMinutiaCapacity Function 344

6.6.1.87. NFRecordSetMinutiaFormat Function 345
6.6.1.88. NFRecordSetMinutiaNeighbour Function 346
6.6.1.89. NFRecordSetPatternClass Function 347
6.6.1.90. NFRecordSetPosition Function 348
6.6.1.91. NFRecordSetQuality Function 348
6.6.1.92. NFRecordSetRidgeCountsType Function 349
6.6.1.93. NFRidgeCountsType Enumeration 350
6.7. NfsMatcher Library 350
6.7.1. NfsMatcher Module 351
6.7.1.1. NfsmCopyParameters Function 353
6.7.1.2. NfsmCreate Function 354
6.7.1.3. NfsmFingerMatchDetails Structure 355
6.7.1.3.1. NfsmFingerMatchDetails.MatchedIndex Field 355
6.7.1.3.2. NfsmFingerMatchDetails.NfmMatchDetails Field 356
6.7.1.3.3. NfsmFingerMatchDetails.Score Field 356
6.7.1.3.4. NfsmFingerMatchDetails.VfmMatchDetails Field 356
6.7.1.4. NfsmFree Function 357
6.7.1.5. NfsmGetParameter Function 357
6.7.1.6. NfsmIdentifyEnd Function 359
6.7.1.7. NfsmIdentifyNext Function 359
6.7.1.8. NfsmIdentifyStart Function 361
6.7.1.9. NfsmIsRegistered Function 362
6.7.1.10. NfsmMatchDetails Structure 362
6.7.1.10.1. NfsmMatchDetails.FingerCount Field 363
6.7.1.10.2. NfsmMatchDetails.Fingers Field 363
6.7.1.10.3. NfsmMatchDetails.Score Field 363
6.7.1.11. NfsmMatchDetailsFree Function 363
6.7.1.12. NfsmRegister Function 364
6.7.1.13. NfsmReset Function 364
6.7.1.14. NfsmSetParameter Function 365
6.7.1.15. NfsmUnregister Function 367
6.7.1.16. NfsmUsedMatcher Enumeration 367
6.7.1.17. NfsmVerify Function 367
6.8. NFTemplate Library 369
6.8.1. NFTemplate Module 369
6.8.1.1. NFTemplateAddRecord Function 371
6.8.1.2. NFTemplateAddRecordCopy Function 373
6.8.1.3. NFTemplateAddRecordFromMemory Function 374
6.8.1.4. NFTemplateCalculateSize Function 375
6.8.1.5. NFTemplateCheck Function 376
6.8.1.6. NFTemplateClearRecords Function 377
6.8.1.7. NFTemplateClone Function 377
6.8.1.8. NFTemplateCreate Function 378
6.8.1.9. NFTemplateCreateFromMemory Function 379
6.8.1.10. NFTemplateFree Function 380
6.8.1.11. NFTemplateGetRecord Function 381
6.8.1.12. NFTemplateGetRecordCapacity Function 382
6.8.1.13. NFTemplateGetRecordCount Function 382
6.8.1.14. NFTemplateGetRecordCountMem Function 383
6.8.1.15. NFTemplateGetSize Function 384

6.8.1.16. NFTemplatePack Function 385
6.8.1.17. NFTemplateRemoveRecord Function 387
6.8.1.18. NFTemplateSaveToMemory Function 387
6.8.1.19. NFTemplateSetRecordCapacity Function 389
6.8.1.20. NFTemplateUnpack Function 390
6.9. NImages Pro Library 391
6.9.1. Bmp Module 393
6.9.1.1. BmpLoadImageFromFile Function 393
6.9.1.2. BmpLoadImageFromHBitmap Function 394
6.9.1.3. BmpLoadImageFromMemory Function 395
6.9.1.4. BmpSaveImageToFile Function 396
6.9.1.5. BmpSaveImageToHBitmap Function 396
6.9.1.6. BmpSaveImageToMemory Function 397
6.9.2. IHead Module 398
6.9.2.1. IHeadLoadImageFromFile Function 399
6.9.2.2. IHeadLoadImageFromMemory Function 400
6.9.2.3. IHeadSaveImageToFile Function 401
6.9.2.4. IHeadSaveImageToMemory Function 401
6.9.3. NGrayscaleImage Module 402
6.9.3.1. NGrayscaleImageGetPixel Function 403
6.9.3.2. NGrayscaleImageSetPixel Function 404
6.9.4. NImageFile Module 405
6.9.4.1. NImageFileClose Function 406
6.9.4.2. NImageFileCreate Function 406
6.9.4.3. NImageFileFree Function 407
6.9.4.4. NImageFileGetFormat Function 408
6.9.4.5. NImageFileIsOpened Function 409
6.9.4.6. NImageFileReadImage Function 409
6.9.5. NImageFormat Pro Module 410
6.9.5.1. NImageFormatCanRead Function 412
6.9.5.2. NImageFormatCanWrite Function 412
6.9.5.3. NImageFormatCanWriteMultiple Function 413
6.9.5.4. NImageFormatGetBmp Function 414
6.9.5.5. NImageFormatGetDefaultFileExtension Function 414
6.9.5.6. NImageFormatGetFileFilter Function 415
6.9.5.7. NImageFormatGetFormat Function 416
6.9.5.8. NImageFormatGetFormatCount Function 417
6.9.5.9. NImageFormatGetIHead Function 417
6.9.5.10. NImageFormatGetName Function 418
6.9.5.11. NImageFormatGetTiff Function 419
6.9.5.12. NImageFormatGetWsq Function 419
6.9.5.13. NImageFormatLoadImageFromFile Function 420
6.9.5.14. NImageFormatLoadImageFromMemory Function 421
6.9.5.15. NImageFormatOpenFile Function 422
6.9.5.16. NImageFormatOpenFileFromMemory Function 423
6.9.5.17. NImageFormatSaveImageToFile Function 424
6.9.5.18. NImageFormatSaveImageToMemory Function 425
6.9.5.19. NImageFormatSaveImagesToFile Function 426
6.9.5.20. NImageFormatSaveImagesToMemory Function 427
6.9.5.21. NImageFormatSelect Function 428

Copyright © 2006 Neurotechnologija

6.9.6. NImage Pro Module 429
6.9.6.1. NImageClone Function 430
6.9.6.2. NImageCreate Function 431
6.9.6.3. NImageCreateFromData Function 433
6.9.6.4. NImageCreateFromFile Function 435
6.9.6.5. NImageCreateFromImage Function 436
6.9.6.6. NImageCreateFromImageEx Function 437
6.9.6.7. NImageCreateWrapper Function 438
6.9.6.8. NImageFree Function 440
6.9.6.9. NImageGetHeight Function 441
6.9.6.10. NImageGetHorzResolution Function 441
6.9.6.11. NImageGetPixelFormat Function 442
6.9.6.12. NImageGetPixels Function 443
6.9.6.13. NImageGetSize Function 444
6.9.6.14. NImageGetStride Function 445
6.9.6.15. NImageGetVertResolution Function 445
6.9.6.16. NImageGetWidth Function 446
6.9.6.17. NImageSaveToFile Function 447
6.9.7. NImages Pro Module 448
6.9.7.1. NImagesGetGrayscaleColorWrapper Function 448
6.9.7.2. NImagesIsRegistered Function 449
6.9.7.3. NImagesRegister Function 450
6.9.7.4. NImagesUnregister Function 451
6.9.8. NMonochromeImage Module 451
6.9.8.1. NMonochromeImageGetPixel Function 452
6.9.8.2. NMonochromeImageSetPixel Function 453
6.9.9. NPixelFormat Module 454
6.9.9.1. NPixelFormat Enumeration 455
6.9.9.2. NRgb Structure 455
6.9.9.2.1. NRgb.Blue Field 456
6.9.9.2.2. NRgb.Green Field 456
6.9.9.2.3. NRgb.Red Field 456
6.9.10. NRgbImage Module 456
6.9.10.1. NRgbImageGetPixel Function 457
6.9.10.2. NRgbImageSetPixel Function 458
6.9.11. Tiff Module 459
6.9.11.1. TiffLoadImageFromFile Function 459
6.9.11.2. TiffLoadImageFromMemory Function 460
6.9.12. Wsq Module 461
6.9.12.1. WsqLoadImageFromFile Function 462
6.9.12.2. WsqLoadImageFromMemory Function 462
6.9.12.3. WsqSaveImageToFile Function 463
6.9.12.4. WsqSaveImageToMemory Function 464
6.10. NMatcher Library 465
6.10.1. NMatcher Module 466
6.10.1.1. NMCopyParameters Function 468
6.10.1.2. NMCreate Function 469
6.10.1.3. NMFree Function 470
6.10.1.4. NMGetParameter Function 470
6.10.1.5. NMIdentifyEnd Function 472

6.10.1.6. NMIdentifyNext Function 472
6.10.1.7. NMIdentifyStart Function 473
6.10.1.8. NMIsRegistered Function 475
6.10.1.9. NMMatchDetails Structure 475
6.10.1.9.1. NMMatchDetails.FingersMatchDetails Field 475
6.10.1.9.2. NMMatchDetails.Score Field 476
6.10.1.10. NMMatchDetailsFree Function 476
6.10.1.11. NMRegister Function 476
6.10.1.12. NMReset Function 477
6.10.1.13. NMSetParameter Function 478
6.10.1.14. NMUnregister Function 479
6.10.1.15. NMVerify Function 479
6.11. NTemplate Library 481
6.11.1. NTemplate Module 482
6.11.1.1. NTemplateAddFingers Function 483
6.11.1.2. NTemplateAddFingersCopy Function 484
6.11.1.3. NTemplateAddFingersFromMemory Function 485
6.11.1.4. NTemplateCalculateSize Function 486
6.11.1.5. NTemplateCheck Function 487
6.11.1.6. NTemplateClear Function 488
6.11.1.7. NTemplateClone Function 488
6.11.1.8. NTemplateCreate Function 489
6.11.1.9. NTemplateCreateFromMemory Function 490
6.11.1.10. NTemplateFree Function 491
6.11.1.11. NTemplateGetFingers Function 492
6.11.1.12. NTemplateGetSize Function 492
6.11.1.13. NTemplatePack Function 493
6.11.1.14. NTemplateRemoveFingers Function 495
6.11.1.15. NTemplateSaveToMemory Function 495
6.11.1.16. NTemplateUnpack Function 497
6.12. VFExtractor Library 498
6.12.1. VFExtractor Module 499
6.12.1.1. VfeCopyParameters Function 501
6.12.1.2. VfeCreate Function 502
6.12.1.3. VfeExtract Function 503
6.12.1.4. VfeExtractFromImage Function 505
6.12.1.5. VfeExtractUnpacked Function 506
6.12.1.6. VfeExtractUnpackedFromImage Function 508
6.12.1.7. VfeFree Function 509
6.12.1.8. VfeGetMaxTemplateSize Function 510
6.12.1.9. VfeGetParameter Function 510
6.12.1.10. VfeIsRegistered Function 512
6.12.1.11. VfeRegister Function 512
6.12.1.12. VfeReset Function 513
6.12.1.13. VfeReturnedImage Enumeration 514
6.12.1.14. VfeSetParameter Function 514
6.12.1.15. VfeUnregister Function 515
6.13. VFMatcher Library 515
6.13.1. VFMatcher Module 516
6.13.1.1. VfmCopyParameters Function 519

6.13.1.2. VfmCreate Function 520
6.13.1.3. VfmFree Function 521
6.13.1.4. VfmGetParameter Function 521
6.13.1.5. VfmIdentifyEnd Function 522
6.13.1.6. VfmIdentifyNext Function 523
6.13.1.7. VfmIdentifyStart Function 524
6.13.1.8. VfmIsRegistered Function 525
6.13.1.9. VfmMatchDetails Structure 526
6.13.1.9.1. VfmMatchDetails.MatedMinutiaCount Field 526
6.13.1.9.2. VfmMatchDetails.MatedMinutiae Field 526
6.13.1.9.3. VfmMatchDetails.Rotation Field 527
6.13.1.9.4. VfmMatchDetails.Score Field 527
6.13.1.9.5. VfmMatchDetails.TranslationX Field 527
6.13.1.9.6. VfmMatchDetails.TranslationY Field 528
6.13.1.10. VfmMatchDetailsFree Function 528
6.13.1.11. VfmRegister Function 528
6.13.1.12. VfmReset Function 529
6.13.1.13. VfmSetParameter Function 530
6.13.1.14. VfmSpeed Enumeration 531
6.13.1.15. VfmUnregister Function 531
6.13.1.16. VfmVerify Function 532
7. Reference (.NET) 534
7.1. Neurotec Library 535
7.1.1. Neurotec Namespace 535
7.1.1.1. NByteOrder Enumeration 536
7.1.1.2. NIndexPair Structure 537
7.1.1.2.1. Index1 Property 537
7.1.1.2.2. Index2 Property 537
7.1.1.2.3. NIndexPair Constructor 537
7.1.1.3. NRational Structure 538
7.1.1.3.1. NRational Constructor 538
7.1.1.3.2. Empty Field 539
7.1.1.3.3. Denominator Property 539
7.1.1.3.4. Numerator Property 539
7.1.1.4. NURational Structure 539
7.1.1.4.1. NURational Constructor 540
7.1.1.4.2. Empty Field 540
7.1.1.4.3. Denominator Property 540
7.1.1.4.4. Numerator Property 540
7.1.1.5. NeurotecException Class 541
7.1.1.5.1. Code Property 541
7.1.1.5.2. Message Property 541
7.2. Neurotec.Biometrics.ANTemplate Library 541
7.2.1. Neurotec.Biometrics Namespace 542
7.2.1.1. ANTemplate Class 542
7.2.1.1.1. ANTemplate Constructor 543
7.2.1.1.2. Dispose Method 544
7.2.1.1.3. Save Method 545
7.2.1.1.4. ToNFTemplate Method 545
7.2.1.1.5. ToNTemplate Method 545

7.3. Neurotec.Biometrics.FMRecord Library 545
7.3.1. Neurotec.Biometrics Namespace 546
7.3.1.1. FmrCore Structure 547
7.3.1.1.1. FmrCore Constructor 548
7.3.1.1.2. Angle Property 549
7.3.1.1.3. RawAngle Property 549
7.3.1.1.4. X Property 549
7.3.1.1.5. Y Property 549
7.3.1.1.6. ToString Method 550
7.3.1.2. FmrDelta Structure 550
7.3.1.2.1. FmrDelta Constructor 551
7.3.1.2.2. Angle1 Property 552
7.3.1.2.3. Angle2 Property 552
7.3.1.2.4. Angle3 Property 553
7.3.1.2.5. RawAngle1 Property 553
7.3.1.2.6. RawAngle2 Property 553
7.3.1.2.7. RawAngle3 Property 553
7.3.1.2.8. X Property 554
7.3.1.2.9. Y Property 554
7.3.1.2.10. ToString Method 554
7.3.1.3. FmrFingerView Class 555
7.3.1.3.1. Cores Property 556
7.3.1.3.2. Deltas Property 557
7.3.1.3.3. FingerPosition Property 557
7.3.1.3.4. FingerQuality Property 557
7.3.1.3.5. HasEightNeighbourRidgeCounts Property 557
7.3.1.3.6. HasFourNeighbourRidgeCounts Property 558
7.3.1.3.7. ImpressionType Property 558
7.3.1.3.8. Minutiae Property 558
7.3.1.3.9. MinutiaeEightNeighbours Property 558
7.3.1.3.10. MinutiaeFourNeighbours Property 559
7.3.1.3.11. ViewNumber Property 559
7.3.1.3.12. GetMaxSize Method 559
7.3.1.3.13. ToNFRecord Methods 560
7.3.1.4. FmrFingerView.CoreCollection Class 561
7.3.1.4.1. Capacity Property 562
7.3.1.4.2. Count Property 562
7.3.1.4.3. FmrFingerView.CoreCollection.Item Property 563
7.3.1.4.4. Add Method 563
7.3.1.4.5. Clear Method 563
7.3.1.4.6. CopyTo Method 564
7.3.1.4.7. GetEnumerator Method 564
7.3.1.4.8. Insert Method 564
7.3.1.4.9. RemoveAt Method 565
7.3.1.5. FmrFingerView.DeltaCollection Class 565
7.3.1.5.1. Capacity Property 566
7.3.1.5.2. Count Property 566
7.3.1.5.3. FmrFingerView.DeltaCollection.Item Property 566
7.3.1.5.4. Add Method 567
7.3.1.5.5. Clear Method 567

7.3.1.5.6. CopyTo Method 567
7.3.1.5.7. GetEnumerator Method 568
7.3.1.5.8. Insert Method 568
7.3.1.5.9. RemoveAt Method 568
7.3.1.6. FmrFingerView.MinutiaCollection Class 569
7.3.1.6.1. Capacity Property 570
7.3.1.6.2. Count Property 570
7.3.1.6.3. FmrFingerView.MinutiaCollection.Item Property 570
7.3.1.6.4. Add Method 570
7.3.1.6.5. Clear Method 571
7.3.1.6.6. CopyTo Method 571
7.3.1.6.7. GetEnumerator Method 571
7.3.1.6.8. Insert Method 572
7.3.1.6.9. RemoveAt Method 572
7.3.1.7. FmrFingerView.MinutiaEightNeighboursCollection Class ...572
7.3.1.7.1. Count Property 573
7.3.1.7.2. FmrFingerView.MinutiaEightNeighboursCollection.Item Property 573
7.3.1.7.3. GetEnumerator Method 574
7.3.1.8. FmrFingerView.MinutiaFourNeighboursCollection Class 574
7.3.1.8.1. Count Property 575
7.3.1.8.2. FmrFingerView.MinutiaFourNeighboursCollection.Item Property 575
7.3.1.8.3. GetEnumerator Method 576
7.3.1.9. FmrMinutia Structure 576
7.3.1.9.1. FmrMinutia Constructor 577
7.3.1.9.2. Angle Property 579
7.3.1.9.3. Quality Property 579
7.3.1.9.4. RawAngle Property 579
7.3.1.9.5. Type Property 580
7.3.1.9.6. X Property 580
7.3.1.9.7. Y Property 580
7.3.1.9.8. ToString Method 580
7.3.1.10. FMRecord Class 581
7.3.1.10.1. FMRecord Constructor 582
7.3.1.10.2. CaptureEquipmentCompliance Property 585
7.3.1.10.3. CaptureEquipmentId Property 586
7.3.1.10.4. CbeffProductId Property 586
7.3.1.10.5. FingerViews Property 586
7.3.1.10.6. ResolutionX Property 586
7.3.1.10.7. ResolutionY Property 587
7.3.1.10.8. SizeX Property 587
7.3.1.10.9. SizeY Property 587
7.3.1.10.10. Clone Method 587
7.3.1.10.11. Dispose Method 588
7.3.1.10.12. GetSize Method 588
7.3.1.10.13. Save Method 588
7.3.1.10.14. ToNFTemplate Methods 589
7.3.1.10.15. NTemplate Method 590
7.3.1.11. FMRecord.FingerViewCollection Class 591

Copyright © 2006 Neurotechnologija

7.3.1.11.1. Capacity Property 591
7.3.1.11.2. FMRecord.FingerViewCollection.Item Property 592
7.3.1.11.3. Add Method 592
7.3.1.11.4. AddCopy Method 592
7.4. Neurotec.Biometrics.FPScannerMan Library 593
7.4.1. Neurotec.Biometrics Namespace 593
7.4.1.1. FPScanner Class 594
7.4.1.1.1. ID Property 594
7.4.1.1.2. StartCapturing Method 594
7.4.1.1.3. StartCapturingForOneImage Method 595
7.4.1.1.4. StopCapturing Method 595
7.4.1.1.5. FingerPlaced Event 595
7.4.1.1.6. FingerRemoved Event 595
7.4.1.1.7. ImageScanned Event 596
7.4.1.2. FPScannerEventArgs Class 596
7.4.1.2.1. FPScannerEventArgs Constructor 596
7.4.1.2.2. ScannerID Property 597
7.4.1.3. FPScannerImageScannedEventArgs Class 597
7.4.1.3.1. FPScannerImageScannedEventArgs Constructors 597
7.4.1.3.2. Image Property 597
7.4.1.4. FPScannerMan Class 598
7.4.1.4.1. FPScannerMan Constructor 599
7.4.1.4.2. IsRegistered Property 599
7.4.1.4.3. Scanners Property 599
7.4.1.4.4. Dispose Method 599
7.4.1.4.5. Register Method 600
7.4.1.4.6. Unregister Method 600
7.4.1.4.7. ScannerAdded Event 600
7.4.1.4.8. ScannerRemoved Event 600
7.4.1.5. FPScannerMan.ScannerCollection Class 601
7.4.1.5.1. FPScannerMan.ScannerCollection.Item Property 601
7.4.1.6. FPScannerEventHandler Delegate 602
7.4.1.7. FPScannerImageScannedEventHandler Delegate 602
7.5. Neurotec.Biometrics.Gui.NFView Library 602
7.5.1. Fingerprint view component 603
7.5.1.1. NFView Class 603
7.5.1.1.1. NFView Constructor 605
7.5.1.1.2. AllowHover Property 605
7.5.1.1.3. AllowSelection Property 605
7.5.1.1.4. HoveredMinutiaIndex Property 606
7.5.1.1.5. MinutiaColor Property 606
7.5.1.1.6. Image Property 606
7.5.1.1.7. MinutiaColor Property 606
7.5.1.1.8. ResultImage Property 607
7.5.1.1.9. MinutiaColor Property 607
7.5.1.1.10. SelectedMinutiaIndex Property 607
7.5.1.1.11. ShowMinutiae Property 608
7.5.1.1.12. ShownImage Property 608
7.5.1.1.13. Template Property 608
7.5.1.1.14. Zoom Property 608

7.5.1.1.15. GetMinutiaAtScreenPoint Method 609
7.5.1.1.16. ScreenPointToMintiaPosition Method 609
7.5.1.1.17. HoveredMinutiaIndexChanged Event 609
7.5.1.1.18. ImageChanged Event 609
7.5.1.1.19. MinutiaColorChanged Event 610
7.5.1.1.20. NeighbourMinutiaColorChanged Event 610
7.5.1.1.21. ResultImageChanged Event 610
7.5.1.1.22. ShownImageChanged Event 610
7.5.1.1.23. ShownMinutiaeChanged Event 610
7.5.1.1.24. SelectedMinutiaColorChanged Event 610
7.5.1.1.25. SelectedMinutiaIndexChanged Event 610
7.5.1.1.26. TemplateChanged Event 611
7.5.1.1.27. ZoomChanged Event 611
7.5.1.2. ShownImage Enumeration 611
7.6. Neurotec.Biometrics.NFExtractor Library 611
7.6.1. Neurotec.Biometrics Namespace 611
7.6.1.1. NFExtractor Class 612
7.6.1.1.1. NFExtractor Constructor 614
7.6.1.1.2. ExtractedRidgeCounts Property 614
7.6.1.1.3. IsRegistered Property 615
7.6.1.1.4. Mode Property 615
7.6.1.1.5. ReturnedImage Property 615
7.6.1.1.6. CopyParameters Method 615
7.6.1.1.7. Dispose Method 616
7.6.1.1.8. ExtractUnpacked Method 616
7.6.1.1.9. Extract Methods 618
7.6.1.1.10. GetMaxTemplateSize Method 619
7.6.1.1.11. GetParameter Method 620
7.6.1.1.12. GetStaticParameter Method 620
7.6.1.1.13. Register Method 621
7.6.1.1.14. Reset Method 621
7.6.1.1.15. SetParameter Method 621
7.6.1.1.16. SetStaticParameter Method 622
7.6.1.1.17. Unregister Method 622
7.6.1.1.18. UpdateTemplate Method 623
7.6.1.2. NfeReturnedImage Enumeration 625
7.7. Neurotec.Biometrics.NFMatcher Library 625
7.7.1. Neurotec.Biometrics Namespace 626
7.7.1.1. NFMatcher Class 626
7.7.1.1.1. NFMatcher Constructor 629
7.7.1.1.2. IsRegistered Property 629
7.7.1.1.3. MatchingSpeed Property 629
7.7.1.1.4. MatchingThreshold Property 629
7.7.1.1.5. MaximalRotation Property 630
7.7.1.1.6. Mode Property 630
7.7.1.1.7. CopyParameters Method 630
7.7.1.1.8. Dispose Method 631
7.7.1.1.9. GetParameter Method 631
7.7.1.1.10. GetStaticParameter Method 631
7.7.1.1.11. IdentifyEnd Method 632

7.7.1.1.12. IdentifyNext Method 632
7.7.1.1.13. IdentifyStart Method 635
7.7.1.1.14. Register Method 636
7.7.1.1.15. Reset Method 637
7.7.1.1.16. SetParameter Method 637
7.7.1.1.17. SetStaticParameter Method 638
7.7.1.1.18. Unregister Method 638
7.7.1.1.19. Verify Method 638
7.7.1.2. NfmMatchDetails Class 641
7.7.1.2.1. MatedMinutiae Property 642
7.7.1.2.2. Score Property 642
7.7.1.2.3. Dispose Method 642
7.7.1.3. NfmSpeed Enumeration 642
7.8. Neurotec.Biometrics.NFRecord Library 643
7.8.1. Neurotec.Biometrics Namespace 643
7.8.1.1. NFRecord Class 644
7.8.1.1.1. NFRecord Constructors 647
7.8.1.1.2. Cores Property 649
7.8.1.1.3. Deltas Property 650
7.8.1.1.4. DoubleCores Property 650
7.8.1.1.5. G Property 650
7.8.1.1.6. Handle Property 651
7.8.1.1.7. Height Property 651
7.8.1.1.8. HorzResolution Property 651
7.8.1.1.9. ImpressionType Property 651
7.8.1.1.10. Minutiae Property 652
7.8.1.1.11. MinutiaeNeighbours Property 652
7.8.1.1.12. MinutiaFormat Property 652
7.8.1.1.13. PatternClass Property 652
7.8.1.1.14. Position Property 653
7.8.1.1.15. Quality Property 653
7.8.1.1.16. RidgeCountsType Property 653
7.8.1.1.17. VertResolution Property 654
7.8.1.1.18. Width Property 654
7.8.1.1.19. Check Methods 654
7.8.1.1.20. Clone Method 655
7.8.1.1.21. Dispose Method 655
7.8.1.1.22. FromHandle Method 655
7.8.1.1.23. GetG Method 656
7.8.1.1.24. GetHeight Method 656
7.8.1.1.25. GetHorzResolution Method 657
7.8.1.1.26. GetImpressionType Method 657
7.8.1.1.27. GetMaxSize Method 658
7.8.1.1.28. GetMaxSizeV1 Method 659
7.8.1.1.29. GetPatternClass Method 660
7.8.1.1.30. GetPosition Method 661
7.8.1.1.31. GetQuality Method 661
7.8.1.1.32. GetSize Methods 661
7.8.1.1.33. GetSizeV1 Method 662
7.8.1.1.34. GetVertResolution Method 663

Copyright © 2006 Neurotechnologija

7.8.1.1.35. GetWidth Method 664
7.8.1.1.36. Save Methods 664
7.8.1.1.37. SaveV1 Method 665
7.8.1.2. NFRecord.CoreCollection Class 666
7.8.1.2.1. Capacity Property 667
7.8.1.2.2. Count Property 667
7.8.1.2.3. NFRecord.CoreCollection.Item Property 667
7.8.1.2.4. Add Method 668
7.8.1.2.5. Clear Method 668
7.8.1.2.6. CopyTo Method 669
7.8.1.2.7. GetEnumerator Method 669
7.8.1.2.8. Insert Method 669
7.8.1.2.9. RemoveAt Method 670
7.8.1.3. NFRecord.DeltaCollection Class 670
7.8.1.3.1. Capacity Property 671
7.8.1.3.2. Count Property 671
7.8.1.3.3. NFRecord.DeltaCollection.Item Property 671
7.8.1.3.4. Add Method 672
7.8.1.3.5. Clear Method 672
7.8.1.3.6. CopyTo Method 672
7.8.1.3.7. GetEnumerator Method 673
7.8.1.3.8. Insert Method 673
7.8.1.3.9. RemoveAt Method 673
7.8.1.4. NFRecord.DoubleCoreCollection Class 674
7.8.1.4.1. Capacity Property 675
7.8.1.4.2. Count Property 675
7.8.1.4.3. Item Property 675
7.8.1.4.4. Add Method 676
7.8.1.4.5. Clear Method 676
7.8.1.4.6. CopyTo Method 676
7.8.1.4.7. GetEnumerator Method 677
7.8.1.4.8. Insert Method 677
7.8.1.4.9. RemoveAt Method 677
7.8.1.5. NFRecord.MinutiaCollection Class 678
7.8.1.5.1. Capacity Property 678
7.8.1.5.2. Count Property 679
7.8.1.5.3. MinutiaCollection.Item Property 679
7.8.1.5.4. Add Method 679
7.8.1.5.5. Clear Method 680
7.8.1.5.6. CopyTo Method 680
7.8.1.5.7. GetEnumerator Method 680
7.8.1.5.8. Insert Method 681
7.8.1.5.9. RemoveAt Method 681
7.8.1.6. NFRecord.MinutiaNeighboursCollection Class 681
7.8.1.6.1. Count Property 682
7.8.1.6.2. NFRecord.MinutiaNeighboursCollection.Item Property 682
7.8.1.6.3. GetCount Method 683
7.8.1.6.4. GetEnumerator Method 683
7.8.1.7. NFCore Struct 684
7.8.1.7.1. NFCore Constructor 684

7.8.1.7.2. Angle Property 685
7.8.1.7.3. RawAngle Property 685
7.8.1.7.4. X Property 686
7.8.1.7.5. Y Property 686
7.8.1.8. NFDelta Struct 686
7.8.1.8.1. NFDelta Constructor 687
7.8.1.8.2. Angle1 Property 688
7.8.1.8.3. Angle2 Property 689
7.8.1.8.4. Angle3 Property 689
7.8.1.8.5. RawAngle1 Property 689
7.8.1.8.6. RawAngle2 Property 690
7.8.1.8.7. RawAngle3 Property 690
7.8.1.8.8. X Property 690
7.8.1.8.9. Y Property 690
7.8.1.9. NFDoubleCore Struct 691
7.8.1.9.1. NFDoubleCore Constructor 691
7.8.1.9.2. X Property 691
7.8.1.9.3. Y Property 692
7.8.1.10. NFMinutia Struct 692
7.8.1.10.1. NFMinutia Constructor 692
7.8.1.10.2. Angle Property 695
7.8.1.10.3. Curvature Property 695
7.8.1.10.4. G Property 695
7.8.1.10.5. Quality Property 695
7.8.1.10.6. RawAngle Property 696
7.8.1.10.7. Type Property 696
7.8.1.10.8. X Property 696
7.8.1.10.9. Y Property 696
7.8.1.11. NFMinutiaNeighbour Struct 697
7.8.1.11.1. NFMinutiaNeighbour Constructor 697
7.8.1.11.2. Empty Field 698
7.8.1.11.3. Index Property 698
7.8.1.11.4. RidgeCount Property 698
7.8.1.12. NFImpressionType Enumeration 698
7.8.1.13. NFMinutiaFormat Enumeration 699
7.8.1.14. NFMinutiaType Enumeration 699
7.8.1.15. NFPatternClass Enumeration 700
7.8.1.16. NFPosition Enumeration 701
7.8.1.17. NFRidgeCountsType Enumeration 701
7.9. Neurotec.Biometrics.NfsMatcher Library 702
7.9.1. Neurotec.Biometrics Namespace 702
7.9.1.1. NfsmFingerMatchDetails Class 703
7.9.1.1.1. MatchedIndex Property 703
7.9.1.1.2. NfmMatchDetails Property 704
7.9.1.1.3. Score Property 704
7.9.1.1.4. VfmMatchDetails Property 704
7.9.1.1.5. Dispose Method 704
7.9.1.2. NfsmMatchDetails Class 705
7.9.1.2.1. Fingers Property 705
7.9.1.2.2. Score Property 705

7.9.1.3. NfsmMatchDetails.NfsmFingerMatchDetailsCollection Class 705
7.9.1.3.1. Count Property 706
7.9.1.3.2. NfsmMatchDetails.NfsmFingerMatchDetailsCollection.Item Property 706
7.9.1.3.3. Contains Method 707
7.9.1.3.4. GetEnumerator Method 707
7.9.1.3.5. IndexOf Method 707
7.9.1.4. NfsMatcher Class 708
7.9.1.4.1. NfsMatcher Constructor 710
7.9.1.4.2. IsRegistered Property 710
7.9.1.4.3. MatchingThreshold Property 711
7.9.1.4.4. NfmMatchingSpeed Property 711
7.9.1.4.5. NfmMaximalRotation Property 711
7.9.1.4.6. NfmMode Property 711
7.9.1.4.7. UsedFingerMatcher Property 712
7.9.1.4.8. VfmMatchingSpeed Property 712
7.9.1.4.9. VfmMaximalRotation Property 712
7.9.1.4.10. VfmMode Property 712
7.9.1.4.11. CopyParameters Method 713
7.9.1.4.12. Dispose Method 713
7.9.1.4.13. GetParameter Method 713
7.9.1.4.14. GetStaticParameter Method 714
7.9.1.4.15. IdentifyEnd Method 715
7.9.1.4.16. IdentifyNext Method 715
7.9.1.4.17. IdentifyStart Method 718
7.9.1.4.18. Register Method 720
7.9.1.4.19. Reset Method 720
7.9.1.4.20. SetParameter Method 720
7.9.1.4.21. SetStaticParameter Method 721
7.9.1.4.22. Unregister Method 722
7.9.1.4.23. Verify Method 722
7.9.1.5. NfsmUsedMatcher Enumeration 725
7.10. Neurotec.Biometrics.NFTemplate Library 725
7.10.1. Neurotec.Biometrics Namespace 725
7.10.1.1. NFTemplate Class 726
7.10.1.1.1. NFTemplate Constructor 727
7.10.1.1.2. Handle Property 729
7.10.1.1.3. Records Property 729
7.10.1.1.4. CalculateSize Method 730
7.10.1.1.5. Check Method 730
7.10.1.1.6. Clone Method 731
7.10.1.1.7. Dispose Method 731
7.10.1.1.8. FromHandle Method 731
7.10.1.1.9. GetRecordCount Method 732
7.10.1.1.10. GetSize Method 733
7.10.1.1.11. Pack Method 734
7.10.1.1.12. Save Method 735
7.10.1.1.13. Unpack Method 735
7.10.1.2. NFTemplate.RecordCollection Class 739
7.10.1.2.1. Capacity Property 739

Copyright © 2006 Neurotechnologija

7.10.1.2.2. NFTemplate.RecordCollection.Item Property 739
7.10.1.2.3. Add Methods 740
7.10.1.2.4. AddCopy Method 742
7.11. Neurotec.Biometrics.NMatcher Library 742
7.11.1. Neurotec.Biometrics Namespace 743
7.11.1.1. NMatcher Class 743
7.11.1.1.1. NMatcher Constructor 745
7.11.1.1.2. FingersNfmMatchingSpeed Property 745
7.11.1.1.3. FingersNfmMaximalRotation Property 746
7.11.1.1.4. FingersNfmMode Property 746
7.11.1.1.5. FingersUsedMatcher Property 746
7.11.1.1.6. FingersVfmMatchingSpeed Property 746
7.11.1.1.7. FingersVfmMaximalRotation Property 747
7.11.1.1.8. FingersVfmMode Property 747
7.11.1.1.9. IsRegistered Property 747
7.11.1.1.10. MatchingThreshold Property 747
7.11.1.1.11. CopyParameters Method 748
7.11.1.1.12. Dispose Method 748
7.11.1.1.13. GetParameter Method 748
7.11.1.1.14. GetStaticParameter Method 749
7.11.1.1.15. IdentifyEnd Method 750
7.11.1.1.16. IdentifyNext Method 750
7.11.1.1.17. IdentifyStart Method 753
7.11.1.1.18. Register Method 754
7.11.1.1.19. Reset Method 755
7.11.1.1.20. SetParameter Method 755
7.11.1.1.21. SetStaticParameter Method 756
7.11.1.1.22. Unregister Method 756
7.11.1.1.23. Verify Method 757
7.11.1.2. NMMatchDetails Class 760
7.11.1.2.1. FingersMatchDetails Property 760
7.11.1.2.2. Score Property 760
7.11.1.2.3. Dispose Method 761
7.12. Neurotec.Biometrics.NTemplate Library 761
7.12.1. Neurotec.Biometrics Namespace 761
7.12.1.1. NTemplate Class 761
7.12.1.1.1. NTemplate Constructor 762
7.12.1.1.2. Fingers Property 765
7.12.1.1.3. Handle Property 765
7.12.1.1.4. AddFingers Methods 765
7.12.1.1.5. AddFingersCopy Method 766
7.12.1.1.6. CalculateSize Method 767
7.12.1.1.7. Check 767
7.12.1.1.8. Clear Method 768
7.12.1.1.9. Clone Method 769
7.12.1.1.10. Dispose Method 769
7.12.1.1.11. FromHandle Method 769
7.12.1.1.12. GetSize Methods 769
7.12.1.1.13. Pack Methods 770
7.12.1.1.14. RemoveFingers Method 771

Copyright © 2006 Neurotechnologija

7.12.1.1.15. Save Methods 771
7.12.1.1.16. Unpack Methods 772
7.13. Neurotec.Biometrics.VFExtractor Library 775
7.13.1. Neurotec.Biometrics Namespace 776
7.13.1.1. VFExtractor Class 776
7.13.1.1.1. VFExtractor Constructor 778
7.13.1.1.2. IsRegistered Property 778
7.13.1.1.3. Mode Property 779
7.13.1.1.4. ReturnedImage Property 779
7.13.1.1.5. CopyParameters Method 779
7.13.1.1.6. Dispose Method 780
7.13.1.1.7. Extract Methods 780
7.13.1.1.8. ExtractUnpacked Method 782
7.13.1.1.9. GetMaxTemplateSize Method 783
7.13.1.1.10. GetParameter Method 784
7.13.1.1.11. GetStaticParameter Method 784
7.13.1.1.12. Register Method 785
7.13.1.1.13. Reset Method 785
7.13.1.1.14. SetParameter Method 785
7.13.1.1.15. SetStaticParameter Method 786
7.13.1.1.16. Unregister Method 786
7.13.1.2. VfeReturnedImage Enumeration 787
7.14. Neurotec.Biometrics.VFMatcher Library 787
7.14.1. Neurotec.Biometrics Namespace 787
7.14.1.1. VfmMatchDetails Class 788
7.14.1.1.1. MatedMinutiae Property 788
7.14.1.1.2. RawRotation Property 789
7.14.1.1.3. Rotation Property 789
7.14.1.1.4. Score Property 789
7.14.1.1.5. TranslationX Property 789
7.14.1.1.6. TranslationY Property 789
7.14.1.1.7. Dispose Method 789
7.14.1.2. VFMatcher Class 789
7.14.1.2.1. VFMatcher Constructor 792
7.14.1.2.2. IsRegistered Property 792
7.14.1.2.3. MatchingSpeed Property 792
7.14.1.2.4. MatchingThreshold Property 793
7.14.1.2.5. MaximalRotation Property 793
7.14.1.2.6. Mode Property 793
7.14.1.2.7. CopyParameters Method 793
7.14.1.2.8. Dispose Method 794
7.14.1.2.9. GetParameter Method 794
7.14.1.2.10. GetStaticParameter Method 795
7.14.1.2.11. IdentifyEnd Method 795
7.14.1.2.12. IdentifyNext Method 796
7.14.1.2.13. IdentifyStart Method 798
7.14.1.2.14. Register Method 800
7.14.1.2.15. Reset Method 800
7.14.1.2.16. SetParameter Method 800
7.14.1.2.17. SetStaticParameter Method 801

Copyright © 2006 Neurotechnologija

7.14.1.2.18. Unregister Method 802
7.14.1.2.19. Verify Method 802
7.14.1.3. VfmSpeed Enumeration 805
7.15. Neurotec.Images Pro Library 805
7.15.1. Neurotec.Images Namespace 805
7.15.1.1. Bmp Class 806
7.15.1.1.1. LoadImage Method 807
7.15.1.1.2. LoadImageFromBitmap Method 808
7.15.1.1.3. LoadImageFromHBitmap Method 809
7.15.1.1.4. SaveImage Method 810
7.15.1.1.5. SaveImageToBitmap Method 811
7.15.1.1.6. SaveImageToHBitmap Method 811
7.15.1.2. IHead Class 812
7.15.1.2.1. LoadImage Method 812
7.15.1.2.2. SaveImage Method 814
7.15.1.3. NGrayscaleImage Class 815
7.15.1.3.1. NGrayscaleImage.Item Property 815
7.15.1.4. NImage Class 815
7.15.1.4.1. Handle Property 817
7.15.1.4.2. Height Property 817
7.15.1.4.3. HorzResolution Property 817
7.15.1.4.4. LongSize Property 818
7.15.1.4.5. LongStride Property 818
7.15.1.4.6. PixelFormat Property 818
7.15.1.4.7. Pixels Property 819
7.15.1.4.8. Size Property 819
7.15.1.4.9. Stride Property 820
7.15.1.4.10. VertResolution Property 820
7.15.1.4.11. Width Property 820
7.15.1.4.12. Clone Method 821
7.15.1.4.13. Create Method 821
7.15.1.4.14. Dispose Method 823
7.15.1.4.15. FromBitmap Method 823
7.15.1.4.16. FromData Method 823
7.15.1.4.17. FromFile Method 825
7.15.1.4.18. FromHandle Method 826
7.15.1.4.19. FromHBitmap Method 827
7.15.1.4.20. FromImage Method 827
7.15.1.4.21. GetWrapper Method 830
7.15.1.4.22. Save Method 832
7.15.1.4.23. ToBitmap Method 833
7.15.1.4.24. ToHBitmap Method 833
7.15.1.5. NImageFile Class 833
7.15.1.5.1. Format Property 834
7.15.1.5.2. IsOpened Property 834
7.15.1.5.3. Close Method 834
7.15.1.5.4. Dispose Method 835
7.15.1.5.5. FromFile Method 835
7.15.1.5.6. ReadImage Method 836
7.15.1.6. NImageFormat Class 836

Copyright © 2006 Neurotechnologija

7.15.1.6.1. Bmp Field 837
7.15.1.6.2. Formats Field 837
7.15.1.6.3. Gif Field 837
7.15.1.6.4. IHead Field 838
7.15.1.6.5. Jpeg Field 838
7.15.1.6.6. Png Field 838
7.15.1.6.7. Tiff Field 838
7.15.1.6.8. Wsq Field 838
7.15.1.6.9. CanRead Property 838
7.15.1.6.10. CanWrite Property 838
7.15.1.6.11. CanWriteMultiple Property 839
7.15.1.6.12. DefaultFileExtension Property 839
7.15.1.6.13. FileFilter Property 839
7.15.1.6.14. Name Property 839
7.15.1.6.15. LoadImage Method 840
7.15.1.6.16. OpenFile Method 841
7.15.1.6.17. SaveImage Method 842
7.15.1.6.18. SaveImages Method 844
7.15.1.6.19. Select Method 845
7.15.1.7. NImageFormat.ImageFormatCollection Class 846
7.15.1.7.1. ImageFormatCollection.Item Property 846
7.15.1.7.2. IndexOf Method 846
7.15.1.8. NImages Class 847
7.15.1.8.1. IsRegistered Property 847
7.15.1.8.2. GetGrayscaleColorWrapper Method 848
7.15.1.8.3. Register Method 848
7.15.1.8.4. Unregister Method 849
7.15.1.9. NMonochromeImage Class 849
7.15.1.9.1. NMonochromeImage.Item Property 849
7.15.1.10. NPixelFormat Struct 850
7.15.1.10.1. Grayscale Field 851
7.15.1.10.2. Monochrome Field 851
7.15.1.10.3. Rgb Field 852
7.15.1.10.4. BitsPerPixel Property 852
7.15.1.10.5. CalcRowLongSize Methods 852
7.15.1.10.6. CalcRowSize Methods 853
7.15.1.10.7. Equals Method 854
7.15.1.10.8. GetHashCode Method 854
7.15.1.10.9. GetRowLongSize Method 855
7.15.1.10.10. GetRowSize Method 855
7.15.1.10.11. IsValid Method 856
7.15.1.11. NRgb Struct 857
7.15.1.11.1. NTgb constructor 857
7.15.1.11.2. Blue Property 857
7.15.1.11.3. Green Property 858
7.15.1.11.4. Red Property 858
7.15.1.12. NRgbImage Class 858
7.15.1.12.1. NRgbImage.Item Property 858
7.15.1.13. Tiff Class 859
7.15.1.13.1. Tiff Constructor 859

Copyright © 2006 Neurotechnologija

7.15.1.13.2. LoadImage Method 859
7.15.1.14. Wsq Class 861
7.15.1.14.1. LoadImage Method 861
7.15.1.14.2. SaveImage Method 863
A. Support Information 867
B. Distribution Content 868
C. Change Log 877
C.1. ANTemplate Library 880
C.2. Admin Sample 881
C.3. FMRecord Library 881
C.4. FPScannerMan Library 882
C.5. FPScannerManCom Library 882
C.6. FPSmmAtmel Library 882
C.7. FPSmmBiometrika Library 882
C.8. FPSmmCrossMatch Library 883
C.9. FPSmmFM200 Library 883
C.10. FPSmmIdentix Library 883
C.11. FPSmmLighTunning Library 883
C.12. FPSmmST Library 884
C.13. FPSmmTacoma Library 884
C.14. FPSmmUareU Library 884
C.15. Logger 884
C.16. MatchingClusterClient Sample 884
C.17. MySQL Driver 885
C.18. NCore Library 885
C.19. NFExtractor Library 886
C.20. NFMatcher Library 887
C.21. NFRecord Library 887
C.22. NFTemplate Library 888
C.23. NImages Pro Library 888
C.24. NMatcher Library 889
C.25. NTemplate Library 890
C.26. Neurotec Library 890
C.27. Neurotec.Biometrics.ANTemplate Library 891
C.28. Neurotec.Biometrics.FMRecord Library 891
C.29. Neurotec.Biometrics.FPScannerMan Library 891
C.30. Neurotec.Biometrics.Gui.NFView Library 892
C.31. Neurotec.Biometrics.NFExtractor Library 892
C.32. Neurotec.Biometrics.NFMatcher Library 892
C.33. Neurotec.Biometrics.NFRecord Library 893
C.34. Neurotec.Biometrics.NFTemplate Library 893
C.35. Neurotec.Biometrics.NMatcher Library 894
C.36. Neurotec.Biometrics.NTemplate Library 894
C.37. Neurotec.Biometrics.NfsMatcher Library 894
C.38. Neurotec.Biometrics.VFExtractor Library 894
C.39. Neurotec.Biometrics.VFMatcher Library 895
C.40. Neurotec.Images Pro Library 895
C.41. NfsMatcher Library 895
C.42. Node 895
C.43. Server 896

C.44. VFExtractor Library 897
C.45. VFMatcher Library 897
Glossary 898

List of Figures

1.1. The architecture of a Large-Scale Biometric System based on MegaMatcher 2
3.1. Structure of template. 10
4.1. LAN License Manager Monitor 20
4.2. Sending task, 1 scanrio 31
4.3. Sending task, 2 scenario 32
4.4. Database update after template inserting 32
4.5. Requesting information from server 33
4.6. Server - Node communication 44
5.1. Screenshot of Client for .NET sample application 90
5.2. Screenshot of fingerprint record editor window 101

List of Tables

3.1. Information stored in different versions of NFRecord 8
3.2. Optional NFRecord information usage in Neurotechnologija products 8
3.3. Supported scanners for different platforms 11
4.1. Server Configuration Options 23
4.2. Node Configuration Options 26

Chapter 1. About

1.1. Introduction

1.1.1. Large-scale biometric systems in the world

Nowadays it is possible to deal with a complex problem of human identification by his biometric features with the assistance of computers. The complexity of identification task arises from a huge amount of data to be processed. Humans physically cannot accomplish this kind of task, thus automated solutions are required. These solutions are also known as large-scale biometric systems (LSBS).

Many countries incorporate biometric data into passports, ID cards, visas, etc. more often than ever before. Border crossing control, election control, verification of credit cards transactions, traffic control, criminology tasks are time-consuming procedures. Only large-scale biometric systems can handle any of these procedures by matching digitally stored biometric data in fast and reliable manner.

1.1.2. LSBS Requirements

The following common requirements are allocated to large-scale biometric systems:

- Reliability. The system must deal with the identification problem considering huge databases of biometric data. Furthermore, there should be an option to compare fingerprints that have been collected by applying different methods (e.g. comparison of flat and rolled fingerprints). Many organizations have been collecting rolled fingerprints, and they are interested in combining the collected data with other types of fingerprints.
- Productivity and efficiency. This requirement includes the following attributes:
  - Support for huge databases. For example, electors' identification by using database with the citizens' biometric features.
  - Scalability and flexibility. The need for the extension of the system with additional components may arise if the amount of information stored in the database and the number of identification requests increases. These components are intended for increasing the computational power, upgrading existing algorithms, adding new algorithms etc. The system should be upgraded without significant changes in the system architecture.
  - Robustness. The system must be tolerant to software and hardware failures that may interrupt its work.
  - Reaction time. The system must process large amounts of identification requests. Moreover, requests must be processed as soon as possible. The ideal speed of response is real-time.
- Support for biometric standards. The system meeting this requirement could exchange the biometric data with the systems of other vendors without any limitations.
- Remote access. The system must be accessed from any authorized workstation on the network.

Copyright © 2006 Neurotechnologija

1.1.3. MegaMatcher Features

Most of existing large-scale biometric systems are quite expensive. It could be an advantage to introduce a system that whose price is as low as possible. Neurotechnologija offers the SDK that meets all the requirements mentioned above for a competitive price.

Neurotechnologija has developed the MegaMatcher technology (see Figure 1.1, “The architecture of a Large-Scale Biometric System based on MegaMatcher”) that is meets all the requirements set up to large-scale biometric systems.

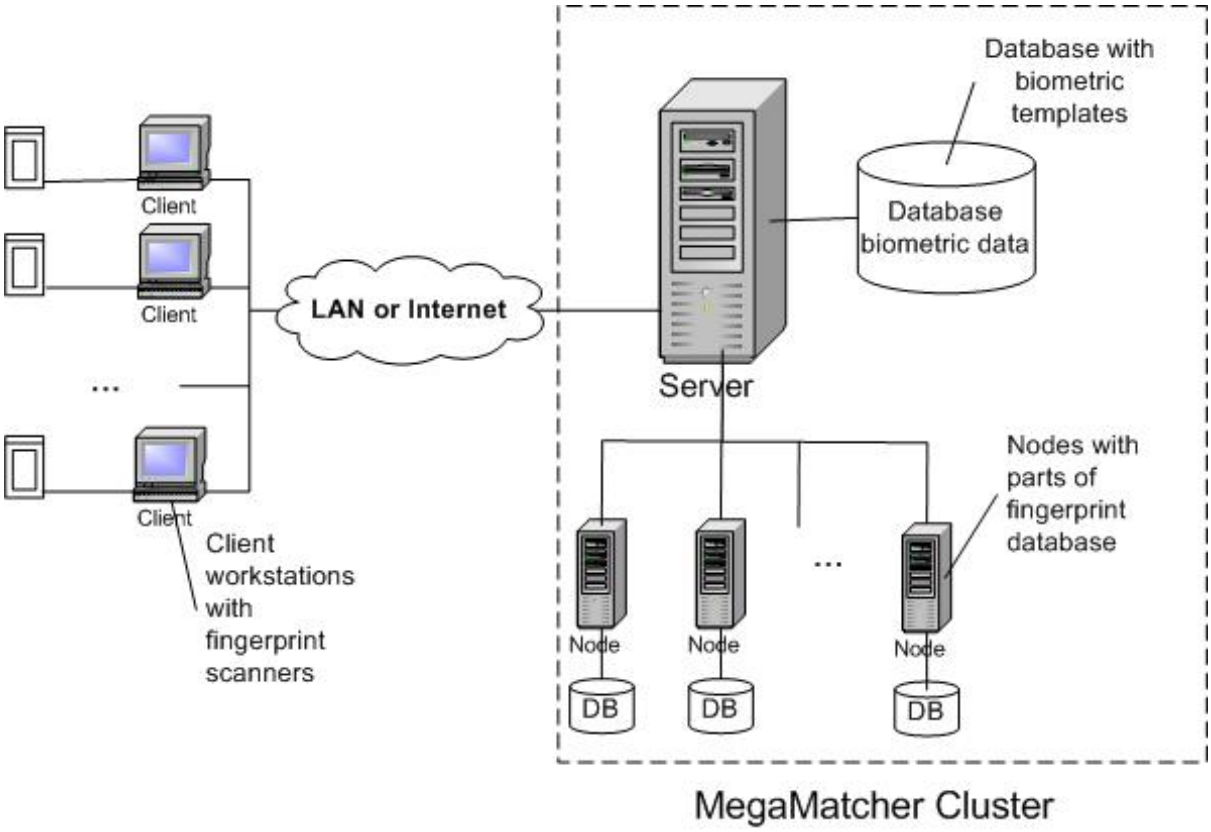

Figure 1.1. The architecture of a Large-Scale Biometric System based on MegaMatcher

The MegaMatcher SDK uses MegaMatcher technology that ensures high reliability of the fingerprint identification and ability to match flat and rolled fingerprints. The MegaMatcher technology includes the computer cluster software that is distributed on the network for performing parallel matching which results in high productivity and efficiency. The location of the Server, Client and Nodes is unconstrained. These parts may be installed optionally on the hardware in the network. The computer cluster is scalable and fault-tolerant that allows performing fast parallel fingerprint matching, processing a large number of identification requests and handling databases with practically unlimited size. The MegaMatcher SDK includes .NET components for rapid development of client side software. The SDK includes WSQ library and some other modules for converting templates of the MegaMatcher to the

templates that are defined by ANSI standards (ANSI/NIST ITL-1-2000 and ANSI/INCITS 378 2004). These modules enable compatibility with other identification systems.

1.2. Platforms Supported

MegaMatcher SDK supports platforms based on x86 processor architecture. Libraries for Windows and Linux operating systems are provided. .NET wrappers of Windows libraries are provided for .NET developers.

1.3. System Requirements

System requirements for installation of the Server Components for Windows OS:

- Computer with x86 compatible CPU (PC with Pentium III 600MHz processor or better is recommended)
- UDP and TCP network support
- Windows 9x/ME/2000/XP/2003 OS
- Oracle Database 10g
- If MySQL choosed as DB engine, MySQL server 4.0 or newer

System requirements for installation of the Server Components for Linux OS:

- Computer with x86 compatible CPU
- UDP and TCP network support
- Linux OS 2.4 or newer
- GCC-3.3.x or newer
- pkg-config-0.15.0 or newer
- GNU Make 3.80 or newer
- Oracle Database 10g
- If MySQL choosed as DB engine, MySQL-4.0.xx or newer client libraries and development packages
- MySQL server 4.0 or newer

System requirements for installation and using of Client Components for Windows OS:

- PC with Pentium III 600MHz processor or better
- Windows 9x/ME/2000/XP/2003 OS
- .NET framework 1.1
- Microsoft Visual Studio .Net 2003
- Oracle Data Provider for .NET 9.2 (for connecting to Oracle from .NET) or newer
- MySQL Connector/Net 1.0 (for connecting to MySQL from .NET) or newer

System requirements for installation and using of Client Components for Linux OS:

- Computer with x86 compatible CPU
- UDP and TCP network support

- Linux OS 2.4 or newer
- GCC-3.3.x or newer
- pkg-config-0.15.0 or newer
- GNU Make 3.80 or newer
- GTK+-2.4.4 or newer libraries and development packages
- libtiff-3.6.x or newer libraries and development packages
- libusb-0.1.8 or newer libraries

System requirements for using of Additional Components for Windows OS:

- PC with Pentium 500MHz processor or better
- Microsoft Windows 9x/ME/NT/2000/XP

System requirements for using of Additional Components for Linux OS:

- Computer with x86 compatible CPU
- UDP and TCP network support
- Linux OS 2.4 or newer
- GCC-3.3.x or newer
- pkg-config-0.15.0 or newer
- GNU Make 3.80 or newer

1.4. Licensing

MegaMatcher SDK has a shared license. This means that you should obtain a license for MegaMatcher SDK once, and then you will have to order licenses only for each component of the SDK. Each license for each component is copy protected by LAN License Manager.

LAN License Manager detects running instances of MegaMatcher components in the LAN. It limits the quantity of simultaneously running instances to the number of licenses stored in the HASP. HASP dongle is a software protection key, manufactured by Aladdin Knowledge Systems Ltd, attached to USB port. LAN License Manager must be installed before running any component of MegaMatcher SDK. The installation of LAN License Manager for MegaMatcher is described in Section 4.1, “LAN License Manager”.

A LAN License Manager Monitor application could be optionally installed on any PC with Windows OS on the same network. It is intended for viewing the complete list of used licenses.

The following components of the MegaMatcher SDK are copy protected by LAN License Manager:

- Server and Matchers of Server Components.
- Fingerprint Features Extraction Components.
- Fingerprint Scanners Support Components.
- Images Support Components.

Chapter 2. What's New

Version 1.1.0.0

- Enables editing Neurotechnologija Finger Records, ANSI INCITS 378-2004 standard templates. NFRecord library interface was changed.
- Cluster Server supports Oracle database, Nodes - SQLite database. The interfaces for joining desirable database was described (Server DB driver). Remote Server, Node logging capability was added. Enabled cluster information retrieval capability to Server about Nodes. The Server, Node configurations options was changed.
- .NET and Visual Basic 6 samples were extended with latent fingerprint record editors. The samples were applied to the changed libraries interfaces.
- NImages Pro library interface was changed and extended. Supports:
  - monochrome and RGB images
  - files with multiple images
  - BMP, TIFF (load-only) and NIST IHead formats
- Supports converting ANSI/NIST-ITL 1-2000 standard files to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates. In detail ANTemplate library.
- FMRecord library interface was changed.
- NfsMatcher, NFMatcher, VFMatcher libraries were separated from NMatcher library.
- NFTemplate library interface was changed.
- NTemplate library interface was changed.

Chapter 3. Overview

The MegaMatcher SDK can be divided into the following parts:

- Server Components.
- Client Components.
- Additional Components.
- Samples.

Person's biometric data (finger print(s), face image(s), etc.) is stored in single storage called a template. Matching could be done against two templates. Suppose there is a database that contains information about persons including the template. Suppose there is a challenge to find a person in such a database having at least a person's template. To do that, the given template must be matched against all those in the database. Due to the possible size of the database (for example, more than 10,000,000 entries), the matching process could last too long. The Cluster comes to solve the problem.

From the very top, the Cluster looks like a system that takes a template containing person's biometric data and searches the database for the person. The result of the search is a list of templates matched against the given template, or affirmation that the given template doesn't match any entry in the database. The Server component of the Cluster uses SQL query mechanisms to split the main database into sub tables, which are distributed to the Nodes. The Node is another component of the Cluster that does actual matching using MegaMatcher engine. From the logical point of view, all Nodes are the same, except that each Node has its own part of the main database. It is obvious, that the bigger the number of Nodes is, the smaller part of the main database is distributed to them and the smaller amount of time is required by matching process to execute. The main requirement for this model is to manage results received from Nodes. The results are managed by the Server and are transmitted to the client on demand. Please note that databases or database management systems are not included into the MegaMatcher SDK.

3.1. Server Components

Server components are:

- Server.
- Node.
- Logger.

The Server program is responsible for main database and tasks transmission to Nodes and matching results management. Server requires database management system software which manages the main database of templates.

Each Node component requires its own database table to store templates. The Node component plays the template role of matcher, so it requires template matching engine which is the part of the MegaMatcher SDK.

Although design of the MegaMatcher SDK does not require direct usage of the matching engine by the developer, its interface is provided in NMatcher module (NMatcher class in .NET) if it is needed.

The Logger program implements a centralized logging system of the Cluster components. Nodes and Server transmit their log messages to the Logger using TCP/IP interface. All the log messages are stored in the single place, so it is easy to monitor the work of the Cluster.

3.2. Client Components

Client components are:

- Fingerprint Features Extraction libraries.
- Template Management libraries.
- Fingerprint Scanner Support libraries.
- Fingerprint View Control library.

3.2.1. Fingerprint Features Extraction

Fingerprint image processing and features extraction is performed with the help of NFExtractor module (NFExtractor class in .NET) to generate fingerprint template that is used during fingerprint verification and/or identification.

The VFExtractor module (VFExtractor class in .NET) is provided to perform fingerprint features extraction using VeriFinger algorithm.

Features extraction can be used with fingerprint images from fingerprint scanners and/or files.

3.2.2. Template Management

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

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

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

Copyright © 2006 Neurotechnologija

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

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

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

Differences between NFRecord versions are summarized in Table 3.1, “Information stored in different versions of NFRecord” and their usage in Table 3.2, “Optional NFRecord information usage in Neurotechnologija products”.

Table 3.1. Information stored in different versions of NFRecord

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

Table 3.2. Optional NFRecord information usage in Neurotechnologija products

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

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

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

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

Figure 3.1, “Structure of template.” shows a brief structure of templates.

Copyright © 2006 Neurotechnologija

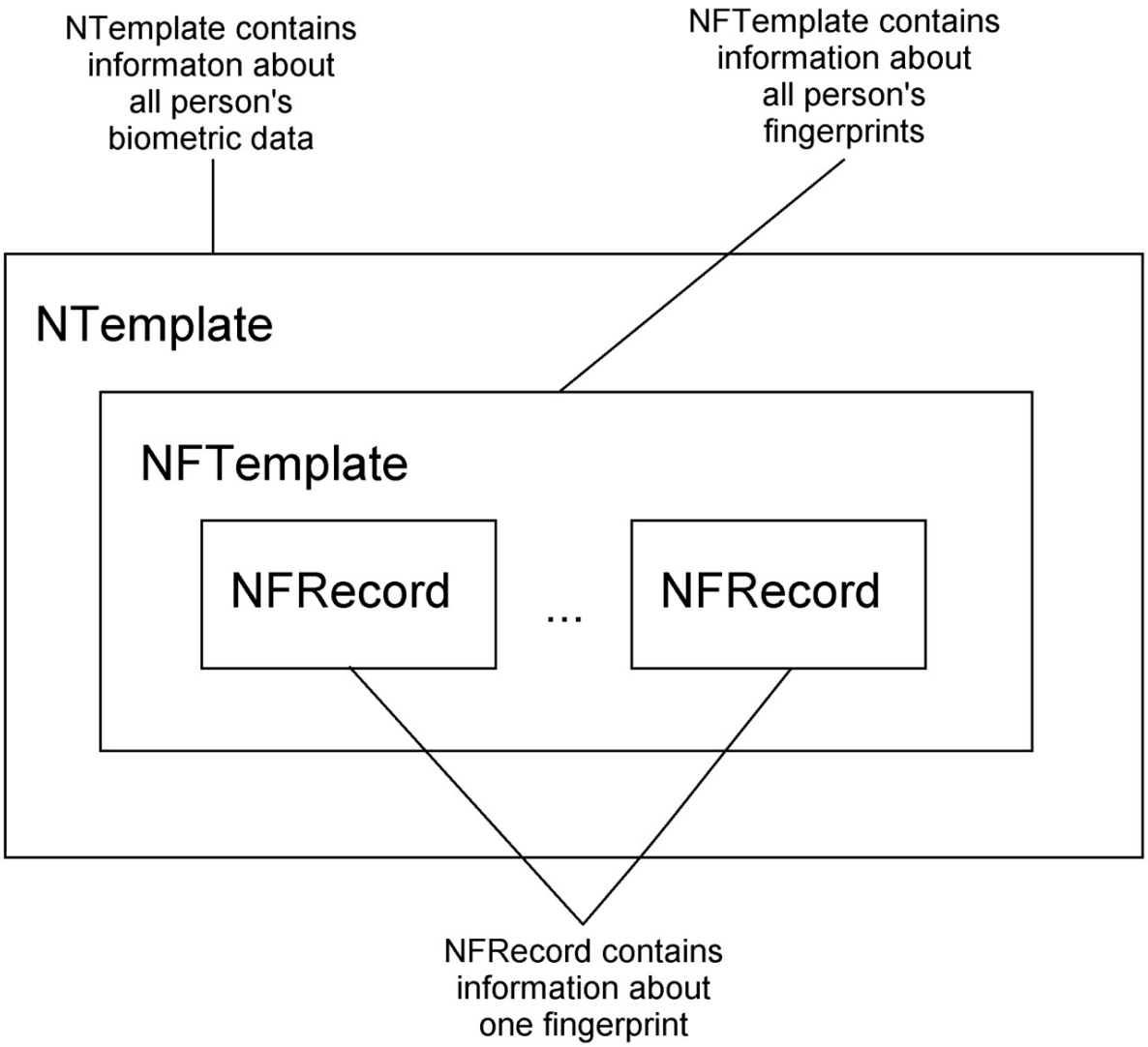

Figure 3.1. Structure of template.

3.2.3. Fingerprint Scanner Support

Fingerprint scanner support allows live scanning of fingerprint for further features extraction. Scanner support is implemented in FPScannerMan class in .NET and in a number of libraries on Linux (Scanners API for Linux OS).

The scanners supported by the MegaMatcher SDK are listed in Table 3.3, “Supported scanners for different platforms”.

Table 3.3. Supported scanners for different platforms

Scanner	.NET	Linux
Atmel FingerChip	x	
BiometriKa FX 2000	x	x
Cross Match Verifier 300-USB	x	x
DigitalPersona U.are.U	x	
Fujitsu MBF200		x
Identix DFR2090	x	
LighTuning LTT-C500	x	
Shimizu/Tacoma CMOS	x	x
Startek FM200	x	x
STMicroelectronics TCRU1C	x	

3.2.3.1. Scanners API for Linux OS

This file has `scanner_info` structure:

```
struct scanner_info
{
int dpi;
char *name;
int version;
void (*close) (void);
int (*init) (void);
unsigned char * (*read) (int *width, int *height);
int timeout;
int max_bad_area;
};
extern struct scanner_info scanner_fx2k;
extern struct scanner_info scanner_shimizu;
extern struct scanner_info scanner_mbf200;
extern struct scanner_info scanner_fm200;
extern struct scanner_info scanner_cross_match;
```

Each scanner driver has an exported name pointing to *`scanner_info`* structure. This structure is the same for every scanner.

scanner_info structure description:

Structure member	Description
int *dpi*	Specifies scanners resolution in DPI.
char **name*	Scanner name in human readable format (null-terminated string).
int *version*	Driver version.
void (**close*) (void)	Function used to close hardware device after use.
int (**init*) (void)	Function used to open hardware device before use. Returns 0 if operation was successful.
unsigned char * (**read*) (int *width, int *height)	Image scanning. Returns *NULL*, if no image could be received from scanner or returns pointer to image if scan was successful. Width and height can't be NULL, returns image size. Returned memory can't be released, and is overwritten with second scan.
int timeout;	User adjustable timeout for USB requests.
int max_bad_area;	The estimation of scanned image is made, according to this value. The interval of this value is from 0 to 100 and it means the amount of non-fingerprint data in the image. The driver compares this value to that calculated for the scanned image and decides if the image is fingerprint. Drivers default for this variable is 85%. The value may be changed at any time during driver execution, depending on user needs. Reasonable values are 65%-85%. NOTE: This option only has sense for Shimizu/Tacoma driver.

Important

These drivers are not thread safe.

Example:

```
#include <stdlib.h>
#include <scanner.h>
struct scanner_info* scanner = &scanner_cross_match;
int main (void)
{
int w, h;
char *image;

// open device
if (scanner->init())
return 1; // initialization error

// scan image
image = scanner->read(&w, &h);
if (image == NULL)
        return 2; // scanning error

// close device
scanner->close();

// return OK
return 0;
}
```

This sample can be compiled using GCC compiler:

```
gcc main.c -Iscanner scanner/cross_match_drv.o -o sample -lusb
```

`cross_match_drv.o` requires `libusb` and must be linked with `-lusb`

`fm200_drv.o` requires `libusb` and must be linked with `-lusb`

`mbf200_drv.o` requires `libusb` and must be linked with `-lusb`

`shimizu_drv.o` requires `libusb` and must be linked with `-lusb`

`fx2k.o` must be linked with `-lfx3L -lfx3scan -lfxoem -lm`

3.2.4. Fingerprint View Control

The fingerprint view control NFView (only for .NET) is used to show a fingerprint image and extracted minutia points.

3.3. Additional Components

Additional components are:

- Image Support library.

- Template Conversion libraries.

3.3.1. Image Support

Image support in the MegaMatcher SDK can be divided into the following four parts:

- Image. The base of all image support. Developers should start using this part and take advantage of other parts if it is required.
- Image Format. Provides a way to learn which image formats are supported and to load and save images in format-neutral way.
- Image File. Should be used if multiple images are stored in one file and more than one image should be loaded from the file.
- Low-Level Image Input-Output. Should be used to have more control on how images are loaded and saved in particular format.

3.3.1.1. Image

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

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

Image in the MegaMatcher SDK is defined by HNImage handle in NImage Pro module (`NImage` class in .NET). It is an encapsulation of a memory block containing image pixels and is organized as rows following each other in top-to-bottom order (number of rows is equal to image height). Each row occupies stride bytes and is organized as pixels following each other in left-to-right order (number of pixels in the row is equal to image width). Each pixel is described by pixel format. See `NImageGetWidth`, `NImageGetHeight`, `NImageGetStride`, `NImageGetPixelFormat` and `NImageGetPixels` functions (Width, Height, Stride, PixelFormat and Pixels properties in .NET) for more information.

An image can have horizontal and vertical resolution attributes assigned to it if they are applicable (they are required for fingerprint image, and do not make sense for face image). See `NImageGetHorzResolution` and `NImageGetVertResolution` functions (HorzResolution and VertResolution properties in .NET) for more information.

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

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

Copyright © 2006 Neurotechnologija

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

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

- On Windows `BmpSaveImageToHBitmap` function (`ToHBitmap` method in .NET) can be used to receive a standard Win32 HBITMAP for the image. The reverse process is also possible using `BmpLoadImageFromHBitmap` function (`FromHBitmap` method in .NET).
- In .NET `ToBitmap` method can be used to receive a standard .NET Bitmap. The reverse process is also possible using `FromBitmap` method.
- On Linux there is no easy method implemented. However, image provides access to memory block containing its pixels via `NImageGetPixelFormat` function (Pixel-Format method in .NET). The memory block can be used to display the image or convert it to some other representation on any platform.

An image can be stored in file in any supported image format using `NImageSaveToFile` function (`Save` method in .NET).

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

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

3.3.1.2. Image Format

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

Image format in the MegaMatcher SDK is defined by HNImageFormat handle in NImage-Format Pro module (`NImageFormat` class in .NET).

There is a number of image formats supported in the MegaMatcher SDK. Not for all of them both reading and writing from/to file is supported on all platforms. See the following table for details.

Image Format	Can read	Can write
BMP	Yes	Yes
GIF	In .NET only	In .NET only
NIST IHead	Yes	Yes
JPEG	In .NET only	In .NET only
PNG	In .NET only	In .NET only
TIFF	Yes	In .NET only

Image Format	Can read	Can write
WSQ	Yes	Yes

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

To learn which image formats are supported in the MegaMatcher SDK in version-independent way `NImageFormatGetFormatCount` and `NImageFormatGetFormat` functions should be used (Formats property in .NET).

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

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

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

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

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

3.3.1.3. Image File

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

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

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

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

3.3.1.4. Low-Level Image Input-Output

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

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

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

3.3.2. Template Conversion

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

- Finger Minutiae Records (FMRecord, Finger Minutiae Format for Data Interchange ANSI/INCITS 378-2004 standard) with the help of FMRecord module (Neurotec.Biometrics.FMRecord class in .NET). This module (class in .NET) also enables reading, writing and editing of FMRecords.
- ANSI/NIST Files (ANTemplate, Data Format for the Interchange of Fingerprint, Facial, & Scar Mark & Tattoo (SMT) Information ANSI/NIST-ITL 1-2000 standard) with the help of ANTemplate module (Neurotec.Biometrics.ANTemplate class in .NET).

3.4. Samples

The following samples are provided with the MegaMatcher SDK:

- The sample of database for Linux OS tool extracts templates from images received from a scanner or file (TIFF, WSQ), and then inserts these templates into the database (see Section 5.1, “Oracle and MySQL database tool for Linux OS”).
- The sample of the server administrating tool for Linux and Windows OS that has a command line interface and several options to show how to administrate the server (see Section 5.2, “Sample of Cluster Administrating tool”).
- Two samples of the client are available:
  - The sample of the client for Linux OS (see Section 5.3, “Client sample for Linux OS”). The command line tool is capable to complete basic tasks in a simple way. The primary purpose of this tool is to demonstrate how the application communicates with the cluster. This tool extracts templates from images received from a scanner or reads them from a file, formats a task batch, sends this batch to the server, gets the completion status of the task, receives the results and removes the results from the server.
  - The sample of the client for .NET platform (see Section 5.4, “Client application for .NET”). This sample demonstrates how the client application receives a fingerprint image from a scanner or a file, enrolls template to a database and communicates with

the cluster. The sample includes cluster communication component which allows sending a task to the cluster, querying its status, getting the results and removing the task from the cluster. This component hides all low-level communications and provides high-level API for the developer.

Copyright © 2006 Neurotechnologija

Chapter 4. Using

4.1. LAN License Manager

MegaMatcher LAN License Manager must be installed before running any part of MegaMatcher SDK. Folders with installation packages of LAN License Manager are:

- for Linux OS `./install/LicenseManager/linux`
- for Windows OS`./install/LicenseManager/windows/`

During the LAN License Manager installation, HASP HL dongle must be plugged into PC, on which LAN License Manager and HASP HL drivers must be installed (\install\LicenseManager\windows\HASP\haspdinst.exe for Windows). To install drivers under Linux OS please follow instructions from `readme.txt` file that is in package ./install/LicenseManager/linux/`HASP_HL.tar.gz`.

LAN License Manager service could be started on any PC that is in the same network as the Cluster. It is recommended to install LAN License Manager on PC that will be used as the server for MegaMatcher cluster.

4.1.1. LAN License Manager Monitor

Also LAN License Manager Monitor could be installed on PC with Windows OS. LAN License Manager Monitor allows to trace how many licenses are in use and update (increase) the quantity of licenses.

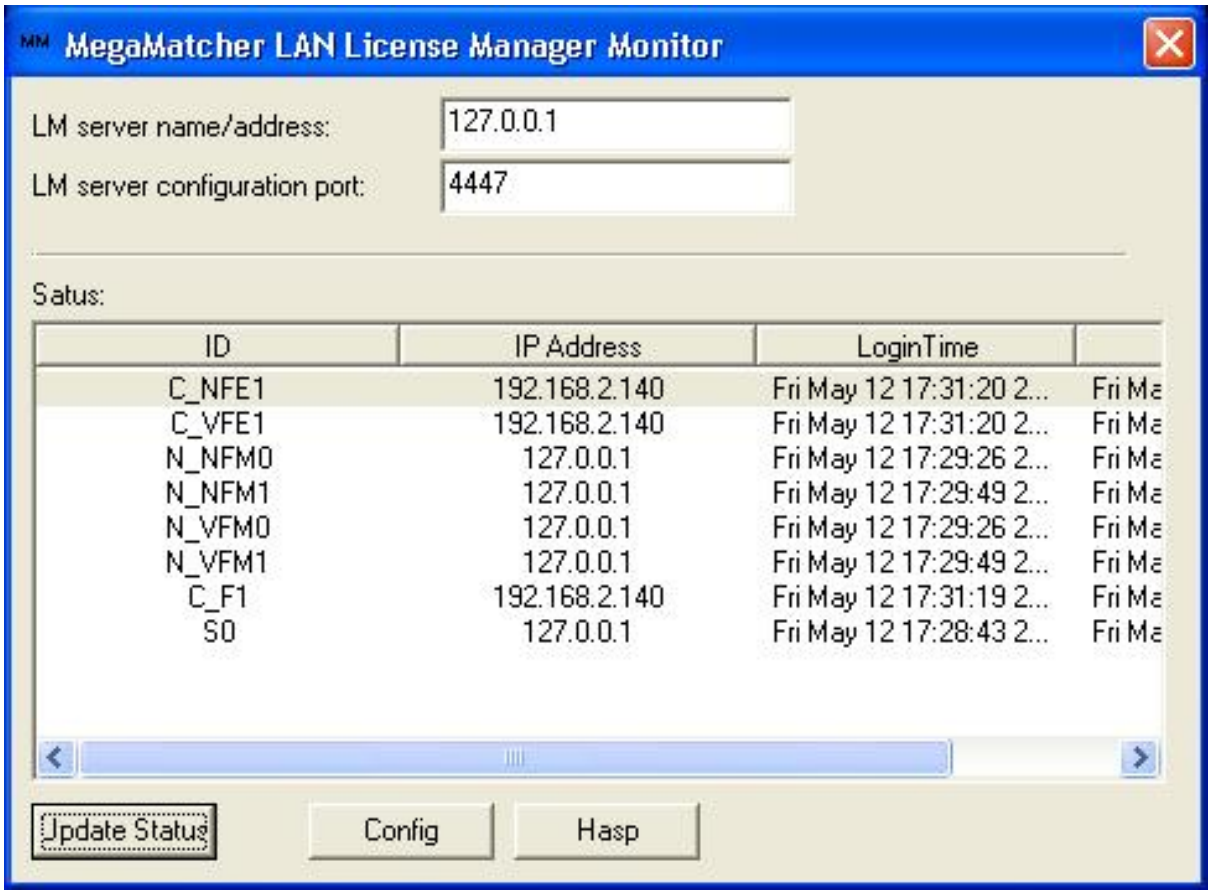

Figure 4.1. LAN License Manager Monitor

4.1.1.1. Main window

MegaMatcher LAN License Manger Monitor window allows you to connect to MegaMatcher LAN License Manager.

- *LM server name/address* - cluster server name/address.
- *LM server configuration port* - cluster server configuration port for connecting to LAN License Manager. Default is 4446.
- *Status* - shows registered libraries and programs. SXXX - server, C_NFEXXX - MegaMatcher extractor from client program, C_VFEXXX - VeriFinger extractor from client program, C_WXXX - WSQ library from client program, C_FXXX - scanner support library from client program, N_NFMXXX - MegaMatcher matcher from node, N_VFMXXX - VeriFinger matcher from node.

4.1.1.2. Configuration window

MegaMatcher LAN License Manger Monitor Configuration window allows you to set MegaMatcher License Manger configuration.

- *Configuration port* - LAN License Manager configuration port to change. Default is 4447
- *Registration port* - libraries and programs registration port. Default is 4446.
- *License timeout* - license timeout in seconds. Default is 300.
- *Config. host* - LAN License Manger IP for joining from LAN License Manager Monitor. It is possible to set particular IP or 255.255.255.255 allows to join from all IPs. Default is 255.255.255.255.

User defined configuration is saved into registries "\HKEY_LOCAL_MACHINE\SOFTWARE\Neurotechnologija\MMLicenseManager"

Important

After changing the LAN License Manager Monitor registration port the server and nodes must be restarted and client application registered with changed ports again.

After restarting LAN License Manager, the server and nodes must be restarted and client application registered again.

4.2. Server Components

4.2.1. Installing

A folder(root folder) should be created for installing the MegaMatcher components. It's recommended to install all files of each component into one folder.

Server

Create a separate folder in MegaMatcher SDK root folder on PC that will be used as a server, and copy these files:

- for Linux OS:
  - `./bin/Linux_x86/server`
  - `./bin/Linux_x86/server.conf`
  - `./lib/Linux_x86/libcommon.so`
  - `./lib/Linux_x86/mysql_drv.so`
  - `./lib/Linux_x86/oracle_drv.so`
- for Windows OS:
  - `/bin/Win32_x86/iconv.dll`
  - `/bin/Win32_x86/intl.dll`
  - `/bin/Win32_x86/msvcr71.dll`
  - `/bin/Win32_x86/mysql_drv.dll`
  - `/bin/Win32_x86/oracle_drv.dll`
  - `/bin/Win32_x86/server.conf`
  - `/bin/Win32_x86/server.exe`
  - `/bin/Win32_x86/libmySQL.dll`

Node

Create a separate folder in MegaMatcher SDK root folder on PC that will be used as a node, and copy these files:

- for Linux OS:
  - `./lib/Linux_x86/libNCore.so`
  - `./lib/Linux_x86/libNFMatcher.so`
  - `./lib/Linux_x86/libNFRecord.so`
  - `./lib/Linux_x86/libNFTemplate.so`
  - `./lib/Linux_x86/libNMatcher.so`
  - `./lib/Linux_x86/libNTemplate.so`
  - `./lib/Linux_x86/libNfsMatcher.so`
  - `./lib/Linux_x86/libVFMatcher.so`
  - `./lib/Linux_x86/libcommon.so`
  - `./bin/Linux_x86/node`
  - `./bin/Linux_x86/node.conf`
- for Windows OS:
  - `/bin/Win32_x86/NCore.dll`
  - `/bin/Win32_x86/NFMatcher.dll`
  - `/bin/Win32_x86/NFRecord.dll`
  - `/bin/Win32_x86/NFTemplate.dll`
  - `/bin/Win32_x86/NMatcher.dll`
  - `/bin/Win32_x86/NTemplate.dll`
  - `/bin/Win32_x86/NfsMatcher.dll`
  - `/bin/Win32_x86/VFMatcher.dll`
  - `/bin/Win32_x86/iconv.dll`
  - `/bin/Win32_x86/intl.dll`
  - `/bin/Win32_x86/msvcr71.dll`
  - `/bin/Win32_x86/node.conf`
  - `/bin/Win32_x86/node.exe`
  - `/bin/Win32_x86/libmySQL.dll`

Logger

Create a separate folder in MegaMatcher SDK root folder on PC that will be used as a Logger, and copy these files:

- for Linux OS:
  - `./lib/Linux_x86/libcommon.so`
  - `./bin/Linux_x86/logger`
  - `./bin/Linux_x86/logger.conf`
- for Windows OS:
  - `/bin/Win32_x86/iconv.dll`
  - `/bin/Win32_x86/intl.dll`
  - `/bin/Win32_x86/logger.exe`
  - `/bin/Win32_x86/logger.conf`
  - `/bin/Win32_x86/msvcr71.dll`

Copyright © 2006 Neurotechnologija

4.2.2. Configuring

The server and nodes are configured using configuration files and command line options. When the Server starts up, it searches for the file named `server.conf` in the current folder, and uses this file for the configuration of the Server. The Node searches for the file named `node.conf` and uses it for the configuration of the Node. These files are called default configuration files.

Some configuration options may be changed using Command Line Interface. The Section 4.2.2.1, “Server” and the Section 4.2.2.2, “Node” describe all the options of configuration files and their values. Default configuration options could be changed by writing a new or changing the existing configuration file. The syntax of configuration file is very simple:

<option name> = <value>

<option name> = <value>

...

Any text editor may be used for writing the configuration file. A new configuration file must be saved with the default configuration filename. A new filename could be specified by using Command Line Interface.

4.2.2.1. Server

All parameters for the configuration of the Server are listed in Table 4.1, “Server Configuration Options”.

Table 4.1. Server Configuration Options

Command Line Option	Conf Option Name	Default Value	Description
-h			Displays short help.
-c		server.conf	<configuration file>
-p <port>	ClientListenPort	25452	TCP port number used by the Server to listen on for the Client connections
-a <address>	ClientListenAdd	0.0.0.0	IP address of NIC to listen for the Client connections
	JoinListenPort	27247	TCP port number used by the Server to listen on for Nodes connections

Copyright © 2006 Neurotechnologija

Command Line Option	Conf Option Name	Default Value	Description
	JoinListenAddr	0.0.0.0	IP address of NIC to listen for Node connections
-l <file name>	LogFileName	server.log	Default Server log filename
-s			This option enables logging to *syslog*.
-o			This option enables server logging to *stdout*. By default, logging to *stdout* is enabled.
	DefaultLogMethod	1	Logging type. The possible values are: 0 - No logging 1 - Log to STDOUT 2 - Log to syslog 4 - Log to file 8 - Log to remote destination which was set using RemoteLoggerAddr parameter To use more than one log destination sum corresponding log destination codes and result set to DefaultLogDest. Example: To log to remote location and file sum corresponding 4 + 8 values and set DefaultLogMethod value to 12
	RemoteLoggerPort	27759	Remote logger port

Command Line Option	Conf Option Name	Default Value	Description
	RemoteLoggerAddr	localhost	Remote logger address
	MonitorListenPort	28015	Port number to listen on for Node health monitor
	MonitorListenAddr	0.0.0.0	IP address of NIC for Node healthy monitor
	AdminListenPort	24932	Port for accepting the administrating tool connections
	AdminListenAddr	0.0.0.0	Default listening address for accepting the administrating tool connections
-u <user>	SQLUser	HR	Authentication user name for the custom database
-w <password>	SQLPassword	HR	Authentication user password for the custom database
-d <database name>	SQLDatabase	//192.168.2.151/XE	Name of the custom database
-r <port>	SQLServerPort	0	Connection port for the custom database
	SQLDBQuery	select * from usertbl	SQL query expression
	SQLDBUpdateQueryPattern	select * from usertbl where dbid = '@id@'	SQL query pattern will be used for database update checking
	DBTemplate	template	Column name of template from the `usertbl` that was included in `SQLDBQuery` expression
	UidName	dbid	Column name for unique ID

Copyright © 2006 Neurotechnologija

Command Line Option	Conf Option Name	Default Value	Description
			(identifier) of template from the *usertbl* that was included in *SQLDBQuery* expression
	NodeCount	2	Number Cluster Nodes. The Cluster will not start if the running Nodes number is less than *NodeCount*
-f <driver name>	SQLDriverName	oracle_drv	SQL driver name that will be used for communication with database. In current version Oracle and MySQL databases are supported. Also desired DB module could be implemented
	NodeCheckInterval	3	An interval in seconds for checking if the Nodes are alive
-P <protection server>	LicenceServer	localhost	Host address or IP address of the PC with the running LAN License Manager

4.2.2.2. Node

All parameters for the configuration of the Node are listed in the Table 4.2, “Node Configuration Options”.

Table 4.2. Node Configuration Options

Command Line Option	Conf Option Name	Default Value	Description
-h			Displays short help.

Command Line Option	Conf Option Name	Default Value	Description
-c <configuration file>		node.conf	Configuration file of Node. Option has higher priority.
-p <port>	ServerPort	27247	TCP port number used to connect to the Server
-s <address>	ServerAddr	127.0.0.1	Host address or IP address of the Server
	MonitorPort	28015	TCP port number of the Server monitor
	MonitorAddr	127.0.0.1	Host address or IP address of the Server monitor
-l <file name>	LogFileName	node.log	The log filename of Node.
-s			This option enables server logging to *syslog*.
-o			This option enables server logging to *stdout*. By default, logging to *stdout* is enabled.
	DefaultLogDest	1	Logging type. The possible values are: 0 - No logging 1 - Log to STDOUT 2 - Log to syslog 4 - Log to file 8 - Log to remote destination which was set using RemoteLoggerAddr parameter To use more than

Command Line Option	Conf Option Name	Default Value	Description
			one log destination sum corresponding log destination codes and result set to DefaultLogDest. Example: To log to remote location and file sum corresponding 4 + 8 values and set DefaultLogMethod value to 12
	RemoteLoggerPort	27759	Remote logger port
	RemoteLoggerAddr	localhost	Remote logger address
-e <host name>	MySQLHost	localhost	Address of PC with a database of Node. It is highly recommended to run the Node database server on the same PC as the Node, because each access to the database is faster when the database server and Node software are located on the same PC **Note** For sqlite database this option has no impact. Sqlite use localhost as host name
	SQLModuleName	sqlite_node_drv	SQL module for node to use. Current version supports SQLite and MySQL node modules
-u <user>	MySQLUser	node	MySQL authentication user name. **Note**

Command Line Option	Conf Option Name	Default Value	Description
			For SQLite DB this option has no impact.
-w <password>	MySQLPassword	node	MySQL authentication user password. **Note** For SQLite DB this option has no impact.
-d <database name>	MySQLDatabase	node_db	SQL database name
-t <table name>	MySQLTable	node_tbl	DB table name. The corresponding tables on all Nodes must have the same name, as it is used in a query sent from the client. Tables are created on the fly, so developer doesn't need to worry about its design
-r <port>	MySQLServerPort	0	MySQL connection port. **Note** For SQLite DB this option has no impact.
-g <mysql_hash_name>	MySQLColumnNameHash	node_hash	Additional column name in which node stores record hash value This column is created on the fly
-i <mysql_id_name>	MySQLColumnNameId	node_id	Additional column name that is associated with the unique record number. This column is created automatically, thus Developer doesn't need to create it in Node DB

Command Line Option	Conf Option Name	Default Value	Description
AlgFileName	for Linux OS: libNMatcher.so for Windows OS: NMatcher.dll	A file name for the matching engine	
-P <protection server>	LicenceServer	localhost	Host or IP address of the PC with running LAN License Manager software.

4.2.3. Running

When the Server starts up it waits for connection with the Nodes. Nodes connect to the Server and register. The registration process is called joining process of Node. When all Nodes have joined the Server, it analyzes the structure of the main table and sends a command to all Nodes to create database tables. Then, during the process of database splitting, Nodes tables are filled with a subset of data that was fetched from the table of the main database. After that the Server is ready and waits for a task from Client. As the task is received, the task ID is generated and sent to all registered Nodes. Client gets the task ID too. When the Node receives the task from the Server it executes a query to get list of templates that will be used for the matching process and the matches given template with templates from the query list. After the matching is done, the Node sends matching results to the Server. The Server collects the results received from the Nodes and saves them, so that they could be sent to the Client on demand. Please, note that many tasks can be sent to the Server and Client may request results for a specific task results using ID of a task.

Server

To start a Server it is enough to:

- for Windows OS: execute `server.exe` file.
- for Linux OS: execute the command `LD_LIBRARY_PATH=./:"path to Oracle Instant client library directory" ./server` from folder, where the Server files where copied.

Note
Oracle client library should be version 10.1.0.4 or less.

Node

To start a Node it is enough to:

- for Windows OS: execute `node.exe` file.
- for Linux OS: execute the command `LD_LIBRARY_PATH=./:"path to sqlite library directory" ./node` from folder, where the Node files where copied.

Logger

To start a Logger it is enough to:

- for Windows OS: execute `loger.exe` file
- for Linux OS: execute the command `LD_LIBRARY_PATH=./ ./loger` from folder, where the Logger files where copied.

4.2.4. Client-Server Communication

Few Client-Server communication scenarios.

4.2.4.1. Sending task and getting results

Client sends task and gets the results.

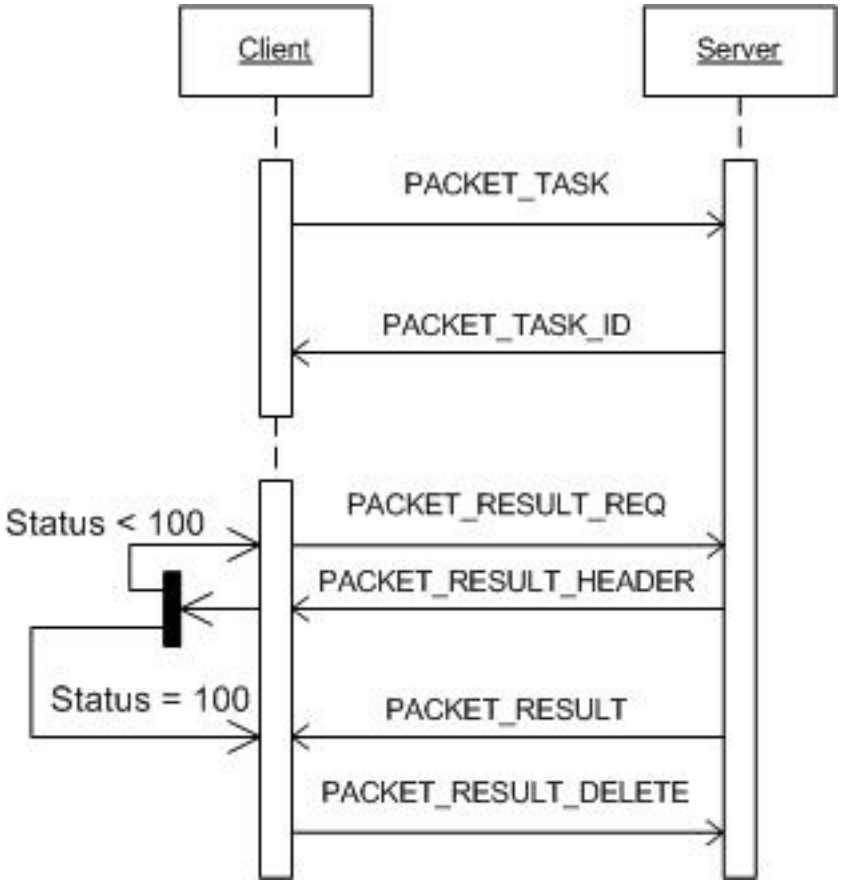

Figure 4.2. Sending task, 1 scanrio

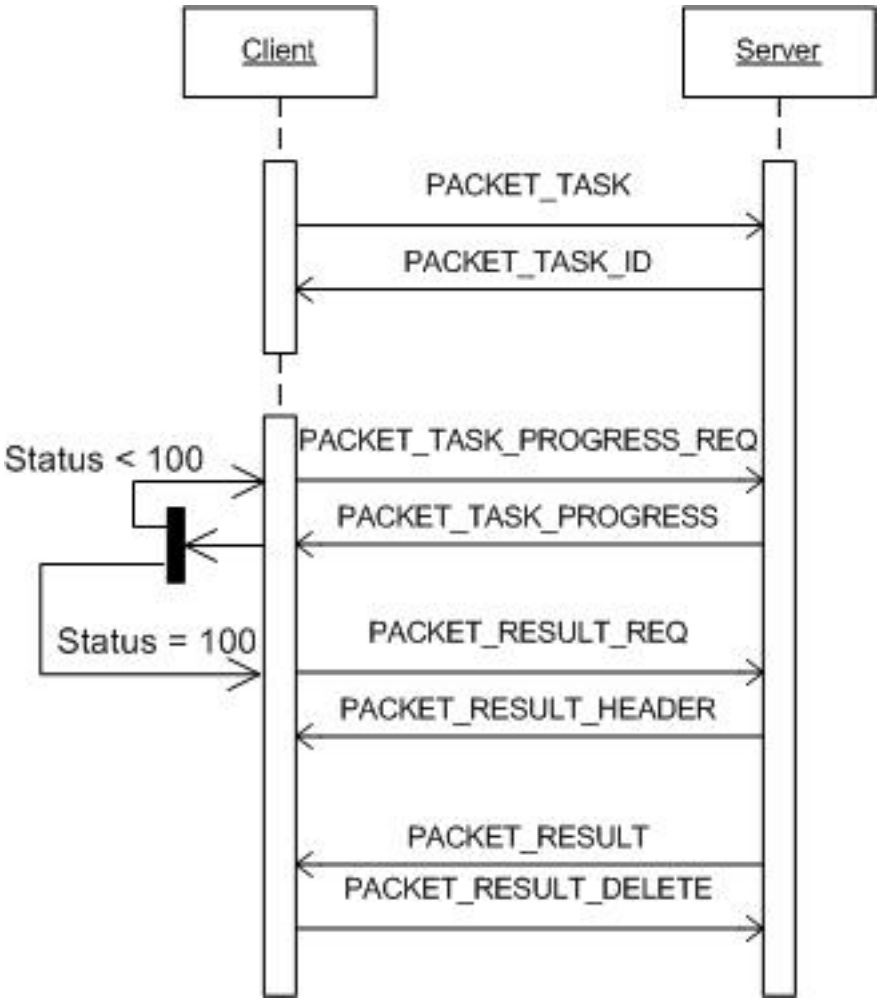

Figure 4.3. Sending task, 2 scenario

4.2.4.2. Inserting template

Client inserts template into database and sends packet to server with template identifier.

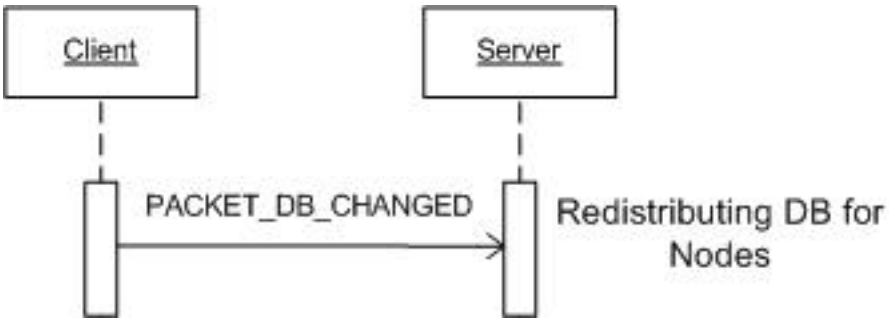

Figure 4.4. Database update after template inserting

4.2.4.3. Getting information

Client requests information from server.

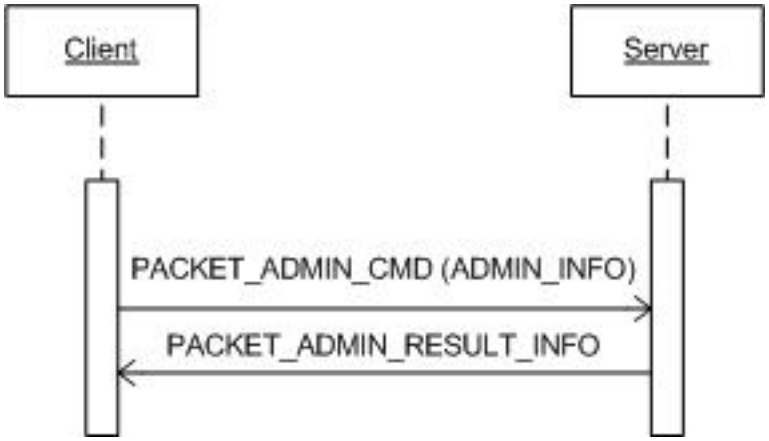

Figure 4.5. Requesting information from server

4.2.5. Packets

4.2.5.1. packet_type Enumeration

```
typedef enum _e_packet_type { } packet_type;
```

Members

PACKET_TASK	The task from client to server.
PACKET_TASK_ID	The task ID from server to client.
PACKET_RESULT_REQ	The result request from client to server.
PACKET_RESULT_HEADER	The result header from client to server.
PACKET_RESULT	The results from server to client.
PACKET_RESULT_DELETE	The result delete from client to server.
PACKET_JOIN	For internal use.
PACKET_JOIN_ACCEPT	For internal use.
PACKET_NODE_MATCH	For internal use.
PACKET_NODE_DBINSERT	For internal use.

PACKET_NODE_DBDELETE	For internal use.
PACKET_NODE_DBCREATE	For internal use.
PACKET_NODE_RESULT_HEADER	For internal use.
PACKET_NODE_RESULT_INFO	For internal use.
PACKET_NODE_ALG_UPDATE	For internal use.
PACKET_NODE_RESULT	For internal use.
PACKET_NODE_FLUSH_TASKS	For internal use.
PACKET_ADMIN_CMD	From client to server.
PACKET_ADMIN_RESULT_INFO	From server to client.
PACKET_DB_CHANGED	From client to server.
PACKET_TASK_PROGRESS_REQ	From client to server.
PACKET_TASK_PROGRESS	From server to client.
PACKET_LOG_MSG	For internal use.

4.2.5.2. PACKET_ADMIN_CMD

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.
Packed Data	4	The command to the server or nodes. One of enumeration adm_type except ADMIN_INFO. The parameter is 0 for server or [1..n] for nodes. If command is ADMIN_INFO, then the parameter should be one of adm_info_type enumeration.
	4	The parameter.

4.2.5.2.1. adm_type Enumeration

Specifies admin command.

```
typedef enum _e_adm_type { } adm_type;
```

Members

ADMIN_START	Start cluster.
ADMIN_STOP	Remove specified node from cluster.
ADMIN_KILL	Stop cluster.
ADMIN_INFO	Get info about cluster status.
ADMIN_DBUPDATE	Split database for all nodes connected to server.
ADMIN_ALGUPDATE	Load new algorithm library on nodes.

4.2.5.2.2. adm_info_type Enumeration

Specifies admin command.

```
typedef enum _e_adm_info_type { } adm_info_type;
```

Members

INFO_RESULTS	The information about pending results.
INFO_NODES	The information about cluster nodes.
INFO_RUNNING_TASKS	The short information about running tasks.
INFO_RUNNING_TASKS_COMPLETE	The complete information about running tasks.

4.2.5.3. PACKET_ADMIN_RESULT_INFO

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.

	Bytes	Description
Packed Data	4	The packet version.
	4	Structures count in data.
	4	Info type is one of adm_info_type.
		Data. One of tructures: node_inf, c_node_state, c_info_r, c_info_inc.

4.2.5.3.1. admin_info Structure

```
typedef struct _admin_info_t { } admin_info;
```

Fields

Type	Name	Description
int	`version`	The structure version.
int	`count`	The number of structures in data.
int	`info_type`	Info type is one of adm_info_type.
char	`data[0]`	One of tructures: node_inf, c_node_state, c_info_r, c_info_inc packed in array.

4.2.5.3.2. node_inf Structure

```
typedef struct _node_inf_t { } node_inf;
```

Fields

Type	Name	Description
int	`node_id`	Node identifier.
int	`progress`	Node task progress.

4.2.5.3.3. c_node_state Structure

```
typedef struct _c_node_state_t { } c_node_state;
```

Fields

Type	Name	Description
int	node_id	Node identifier.
int	state	Node state.

4.2.5.3.4. c_info_r Structure

```
typedef struct _c_info_r_t { } c_info_r;
```

Fields

Type	Name	Description
int	task_id	The task identifier.
int	nodes_completed	Nodes already completed task.
int	w_nodes_count	Amount of nodes working on the task.
int	progress	Task progress.
node_inf	w_nodes_id[0]	Working nodes identifiers.

4.2.5.3.5. c_info_inc Structure

```
typedef struct _c_info_inc_t { } c_info_inc;
```

Fields

Type	Name	Description
int	task_id	The task identifier.
int	nodes_completed	Nodes already completed task.
int	w_nodes_count	Amount of nodes working on the task.

4.2.5.4. PACKET_DB_CHANGED

	Bytes	Description	
Packed Header	4	Packet data size.	
	4	Packet type.	
Packed Data	4	The packet version.	
	4	ID count.	
	4	ID length.	The number of pairs ID length and ID depends on ID count.
		ID which was inserted into DB.	

4.2.5.4.1. db_changed Structure

```
typedef struct _db_changed_t { } db_changed;
```

Fields

Type	Name	Description
int	`version`	The structure version.
int	`count`	The number of template_id structures.
char	`data[0]`	The template_id structures in one array.

4.2.5.4.2. template_id Structure

```
typedef struct _template_id_t { } template_id;
```

Fields

Type	Name	Description
int	`length`	ID length.
char	`id[0]`	ID which was inserted into DB.

4.2.5.5. PACKET_RESULT

	Bytes	Description	
Packed Header	4	Packet data size.	
	4	Packet type.	
Packed Data	4	Matching similarity.	The number of similarity, ID length, ID segments depends on PACKET_RESULT_HEADER count.
	4	Identifier length.	
		Identifier.	

4.2.5.5.1. result Structure

```
typedef struct _result_t { } result;
```

Fields

Type	Name	Description
int	`similarity`	Matching similarity.
int	`id_length`	Identifier length.
char	`id[0]`	Identifier.

4.2.5.6. PACKET_RESULT_DELETE

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.
Packed Data	4	The packet version.
	4	Task identifier.

4.2.5.6.1. result_del Structure

```
typedef struct _result_del_t { } result_del;
```

Fields

Type	Name	Description
int	`version`	The structure version.
int	`task_id`	Task identifier.

4.2.5.7. PACKET_RESULT_HEADER

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.
Packed Data	4	The packet version.
	4	Task completion status. The number from interval [0, 100].
	4	Size of PACKET_RESULT packed data.
	4	The count of similarity, ID length and ID. See PACKET_RESULT packet data.

4.2.5.7.1. result_header Structure

```
typedef struct  { } result_header;
```

Fields

Type	Name	Description
int	`version`	The packet version.
int	`status`	Task completion status. The number from interval [0, 100].
int	`size`	Size of PACKET_RESULT packed data
int	`count`	The count of similarity, ID length and ID.

4.2.5.8. PACKET_RESULT_REQ

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.
Packed Data	4	The packet version.
	4	Task identifier.

4.2.5.8.1. result_req Structure

```
typedef struct _result_req_t { } result_req;
```

Fields

Type	Name	Description
int	`version`	The structure version.
int	`task_id`	Task identifier.

4.2.5.9. PACKET_TASK

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.

	Bytes	Description
Packed Data	4	The packet version.
	4	0 searches all database, 1 searches until first match.
	4	Template size.
	4	Query letght.
	4	Parameters size.
		Template.
		Query for the nodes.
		Parameters.

4.2.5.9.1. task Structure

```
typedef struct _task_t { } task;
```

Fields

Type	Name	Description
int	`version`	The structure version.
task_mode	`mode`	Task mode.
int	`size_template`	Size of template.
int	`size_query`	Size of SQL query.
int	`size_params`	Parameter size.
char	`data[0]`	Template, query, parameters packed into array.

4.2.5.9.2. task_mode Enumeration

```
typedef enum _task_mode_e { } task_mode;
```

Members

`MODE_NORMAL`	Return all results from database.

MODE_FIRST	Return first which match search criteria.

4.2.5.10. PACKET_TASK_ID

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.
Packed Data	4	The packet version.
	4	Task identifier.

4.2.5.10.1. task_id Structure

```
typedef struct _task_id_t { } task_id;
```

Fields

Type	Name	Description
int	version	The structure version.
int	task_id	Task identifier.

4.2.5.11. PACKET_TASK_PROGRESS

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.
Packed Data	4	The packet version.
	4	Task progress.

4.2.5.12. PACKET_TASK_PROGRESS_REQ

	Bytes	Description
Packed Header	4	Packet data size.
	4	Packet type.
Packed Data	4	The packet version.
	4	Task identifier.

4.3. Databases

MegaMatcher SDK supports Oracle and MySQL as Server databases and MySQL and SQLite as node databases. Server uses Server DB Driver to work with specific database. Node uses node DB Module to work with specific database. Database support in cluster can be extended (customized) by implementing (modifying) appropriate DB driver or module.

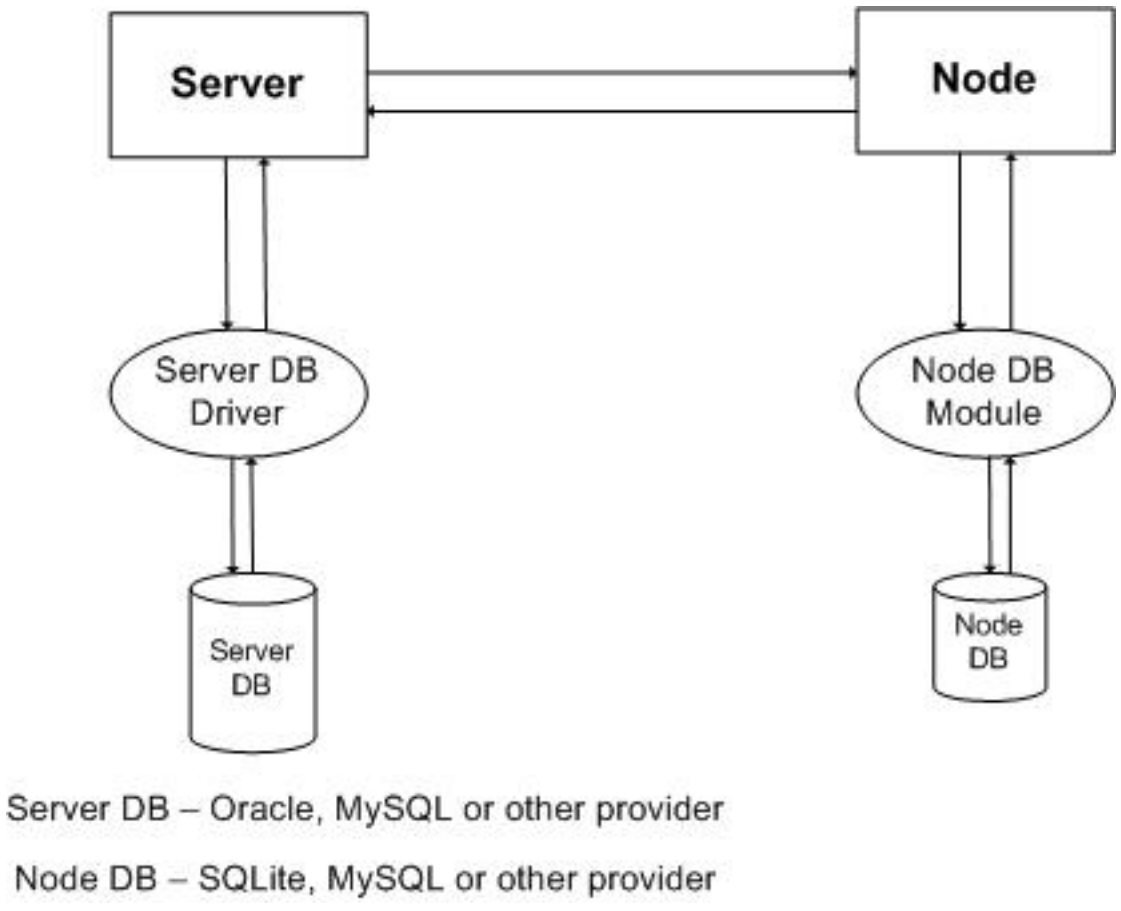

Figure 4.6. Server - Node communication

If specific database functionality is needed in Server - Server DB Driver need to be imple-

mented. If specific database functionality is needed in node - Node DB Module need to be implemented.

By default server uses MySQL DB Driver and node uses SQLite DB Module.

Server and nodes use databases for different purpose and in different manner. Server fetches subset of data table from main DB. This subset can be controlled with *SQLDBQuery* option in `server.conf`. Fetched data must contain templates which are used directly to perform matching, and some field will be used to identify records. This could also include data that later can be used for matching optimizations on nodes. Server doesn't store data. Data is divided (splitted) to nodes immediately after fetch.

Node knows nothing about main DB. When dbsplit event in cluster occurs, server first sends description of fetched data to node. Node uses this data to create his private DB. Then server sends for each node portion of fetched data. Node saves templates in memory, and other data in private DB. When database has been split nodes are ready to perform matching tasks.

Dbsplit event causes server to re fetch and divide whole fetched data. This operation is to costly when main DB undergoes slight changes (one or group of records were inserted, changed or removed). In such occasions server can be asked to re fetch only part of main DB (dbchanged command). Server is given list of identifiers of records what have changed. First server asks nodes to remove data associated with these identifiers. Next it executes query based on *SQLDBUpdateQueryPattern* config option, and sends fetched data to nodes.

One of the parameters of matching task is query. Node can use this query to perform SQL level matching optimizations. For example if gender of fingerprint template owner is known, matching of templates with same gender can be preferred.

4.3.1. MySQL

MySQL database could be used as server and node DB.

4.3.1.1. Installing

MySQL downloads for various operating systems (including Windows and Linux) and instructions on installation could be found at MySQL web site [1]. MySQL database management system could be installed both on the server and on each PC that will be used as a node. If desired SQLite database could be installed on each cluster node instead of MySQL.

4.3.1.2. Configuring

After the Server and Node(-s) are installed and configured, MySQL database server should be configured. The examples below are valid, if default node and server configuration files are used, and both the node and database are running on the same PC.

4.3.1.2.1. Server Configuration

These steps should be completed to create a default database on the server.

[1] http://dev.mysql.com/downloads/

1. Execute MySQL client

2. Create main database that will store templates (the Server and main database could be installed either on the same PC or on different PCs).

```
create database <SQLDatabase>;
```

   Example with the default configuration file:

```
create database userdb;
```

3. Example of main database table.

```
use userdb;

CREATE TABLE usertbl (template blob not null,dbid tinytext not null,
        finger tinyint unsigned not null, scan tinyint unsigned not null,
        user tinytext not null, date datetime not null, size int unsigned not
        null, id int not null AUTO_INCREMENT PRIMARY KEY, g int not null);
```

4. Grant privileges to the server user.

```
grant insert, delete, update, select, drop, create on <SQLDatabase>.* to
<SQLUser>@<MegaMatcherServerHost> identified by '<SQLPassword>';
```

   Example with the default configuration file:

```
grant insert, delete, update, select, drop, create on userdb.* to
userdb@<MegaMatcherServerHost> identified by 'userdb';
```

   Note
   `<MegaMatcherServerHost>` is IP address of the host where MegaMatcher server is running.

Please, note that the database table must have at least 2 fields: first for unique template ID, second for templates data.

4.3.1.2.2. Node(-s) Configuration

These steps are required to create and configure database on each Node of Cluster.

1. Execute MySQL client

2. Create database that will be used by Node.

```
create database <MySQLDatabase>;
```

Example with the default configuration file:

```
create database node_db;
```

3. Grant privileges to the Node user.

```
grant insert, delete, update, select, drop, create on <MySQLDatabase>.* to
<MySQLUser>@<NodeHost> identified by '<SQLPassword>';
```

Note
`<NodeHost>` should be changed with IP address of the host where node is running.

Example with the default configuration file:

```
grant insert, delete, update, select, drop, create on node_db.* to
node@localhost identified by 'node';
```

4.3.1.3. Compiling MySQL DB drivers

Compiled DB drivers are already included in MegaMatcher SDK. This section describes how to recompile them.

4.3.1.3.1. Compiling drivers on Windows

Driver compilation on Windows can be done with help of Microsoft Visual Studio 2003 project file in appropriate source directory. Server DB driver source can be found in `sql_drivers/server_sql_driver/mysql` directory. Node DB driver source can be found in `sql_drivers/node_sql_driver/mysql` directory.

4.3.1.3.2. Compiling server MySQL DB driver on Linux

Server MySQL DB driver need to be compiled using following commands in `sql_drivers/server_sql_driver/mysql` directory:

```
gcc -Wall -I../../../include/linux -I../../include \
`mysql_config --cflags` -O3 -fpic -c -o mysql_drv.o mysql_drv.c

gcc -shared -o mysql_drv.so mysql_drv.o -lcommon -L../../../lib/linux_x86 \
`mysql_config --libs_r`
```

4.3.1.3.3. Compiling and linking node with MySQL DB module on Linux

On Linux node object file `node_object.o` and node DB module object file `mysql_node_drv.o` need to be linked to produce node executable with MySQL support.

First build `mysql_node_drv.o` in `sql_drivers/node_sql_driver/mysql` dir-

ectory:

```
gcc -Wall -I../../../include/linux -I../../include \
`mysql_config --cflags` -O3 -c mysql_db.c -o mysql_node_drv.o
```

Next link node executable in `sql_drivers/node_sql_driver` directory:

```
gcc node_object.o mysql_node_drv.o `mysql_config --libs_r` \
-lcommon -L../../lib/linux_x86 -o node_mysql
```

4.3.2. Oracle

4.3.2.1. Installing

Oracle downloads for various operating systems (including Windows and Linux) and instructions on installation could be found at Oracle web site [2]. For documentation related with Oracle installation and configuration please review Oracle documentation site [3].

To use Oracle DB driver with Server - Instant Oracle client need to be downloaded from Oracle web site [4].

Note

Please use version 10.1.0.4 Instant Oracle client library as version 10.2.0.2 contains some problems which can lead to application crash.

4.3.2.2. Server Configuration

For Oracle database configuration please refer to Oracle documentation. File `create_oracle_table.sql` can be used to create table for default server config (`server.conf`). File `drop_oracle_table.sql` should be used to delete previously created table.

After database installation server config file `server.conf` should be edited to change parameters related with database. Fields *`SQLUser`*, *`SQLPassword`*, *`SQLDatabase`* and other can be modified if needed.

4.3.2.3. Compiling Oracle DB driver

Compiled DB driver is already included in MegaMatcher SDK. This section describes how to recompile it.

Oracle DB driver for Windows can to be compiled using Microsoft Visual Studio 2003 project, which can be found in directory with driver source code (`sql_drivers/server_sql_driver/oracle`).

Linux version of driver can be compiled using following commands in `sql_drivers/server_sql_driver/oracle` directory:

[2] http://www.oracle.com/technology/software/index.html
[3] http://www.oracle.com/technology/documentation/index.html
[4] http://www.oracle.com/technology/software/tech/oci/instantclient/index.html

```
gcc -c -Wall -O3 -fpic -I../../include/ -I <Path to version 10.1.0.4 include files> \
-o oracle_drv.o oracle_drv.c

gcc -shared -O3 -fpic -L <Path to version 10.1.0.4 library files> \
-o oracle_drv.so oracle_drv.o -lclntsh
```

Note
Path should point to Instant Oracle client library or include files.
Note
Please use version 10.1.0.4 Instant Oracle client library as version 10.2.0.2 contains some problems which can lead to application crash.

4.3.3. SQLite

4.3.3.1. Installing

SQLite downloads for various operating systems (including Windows and Linux) and instructions on installation could be found at SQLite web site [5]. SQLite database must be installed on each PC that will be used as a cluster node.

4.3.3.2. Node(-s) configuration

SQLite database do not require any specific configuration steps. When cluster is started only database name and table name is required for SQLite database. Database is created as separate file on node's startup.

4.3.3.3. Compiling SQLite DB driver

Compiled DB driver is already included in MegaMatcher SDK. This section describes how to recompile it.

Node SQLite DB Driver for Windows can be compiled with help of Microsoft Visual Studio 2003 project, which can be found in `sql_drivers/node_sql_driver/sqlite` directory.

On Linux node object file `node_object.o` and node DB module object file `sqlite_node_drv.o` need to be linked to produce node executable.

First build `sqlite_node_drv.o` in `sql_drivers/node_sql_driver/sqlite` directory:

```
gcc -Wall -I/usr/local/include -I../../include/ \
-I../../../include/linux -O3 -c sqlite_db.c -o sqlite_node_drv.o
```

Next link node executable in `sql_drivers/node_sql_driver` directory:

```
gcc node_object.o sqlite/sqlite_node_drv.o -o node -lcommon \
-L../../lib/linux_x86 -lpthread -ltiff -ldl -lm -L/usr/local/lib -lsqlite3
```

[5] http://www.sqlite.org/download.html

4.3.4. Other providers

4.3.4.1. Server DB Driver

This chapter describes functions that need to be implemented in server DB driver.

Functions

connect	DB driver implementation of connection to database method.
close	DB driver implementation of close connection method.
field_count	Return number of SQL query result fields.
fields	Get SQL query result table description.
fetch	Return on row of the results after SQL query execution.
query	DB drivers implementation of query execution method.
query_close	Close query and free internal data structures.
get_object	Create instance of DB driver.
on_delete	Delete instance of DB driver.

4.3.4.1.1. connect function

DB driver implementation of connection to database method.

```
int (SQL_DRV_API *connect)(SQL_driver *driver, char *host, int port,
                           char *username, char *password,
                           char *database);
```

Parameters

driver	[in] Pointer to SQL driver.
host	[in] Pointer to server database host address.
port	[in] Pointer to server database port number.
username	[in] Pointer to server database user name.
password	[in] pointer to server database user password.
database	[in] Pointer to server database name.

Remarks

Server calls this function when it needs to connect to DB server. This function should establish communication with server DB.

Return value

Function returns 0 on success and -1 on error.

4.3.4.1.2. close function

DB driver implementation of close connection method.

```
void (SQL_DRV_API *close)(SQL_driver *driver);
```

Parameters

driver	[in] Pointer to SQL driver.

Remarks

Server calls this function when it needs to close connection with server DB. This function should close connection with server DB.

4.3.4.1.3. field_count function

Return number of SQL query result fields.

```
int (SQL_DRV_API *field_count)(SQL_driver *driver);
```

Parameters

driver	[in] Pointer to SQL driver.

Remarks

Server calls this function when it needs to get number of SQL query result fields after `query` function execution. This function should return number of SQL query result fields.

Return value

Return number of field on success or -1 on error

4.3.4.1.4. fields function

Get SQL query result table description.

```
SQL_field* (SQL_DRV_API *fields)(SQL_driver *driver, int *count);
```

Parameters

driver	[in] Pointer to SQL driver.
count	[in] Pointer to number of fields returned by field_count function.

Remarks

Server calls this function when it needs to get SQL query result columns descriptions. This function should return SQL query results columns descriptions. Specific database SQL types are mapped to generic DB driver SQL types. Column descriptions are transferred to node. These descriptions are used by node DB module to create table on node database. *If some SQL type support is needed in DB Driver; this type should be implemented not only in server DB Driver but also in node DB Module*.

Return value

This function returns SQL query result table description on success and NULL on error.

4.3.4.1.5. fetch function

Return on row of the results after SQL query execution.

```
char** (SQL_DRV_API *fetch)(SQL_driver *driver, unsigned int **sizes);
```

Parameters

driver	[in] Pointer to SQL driver.
sizes	[out] Address of values sizes.

Remarks

Server calls this function when it needs to fetch one row of SQL query result form server DB. This function should fetch one row of SQL query result from server DB. This function should return array of strings which is column values of result row and array of data lengths. NULL values in retrieved results are indicated as -1 in data lengths array. Received result row is send to node, which insert it into node table.

Return value

This function return address of one SQL query result line on success and NULL on error and sizes of result columns. If size of result column equal to -1 then column contains NULL value.

4.3.4.1.6. query function

DB drivers implementation of query execution method.

```
int (SQL_DRV_API *query)(SQL_driver *driver, char *query, int size);
```

Parameters

driver	[in] Pointer to SQL driver.
query	[in] Pointer to query to be executed.
size	[in] Size of the query.

Remarks

Server calls this function when it needs to execute SQL query on server database. This function should execute SQL query specified as *SQLDBQuery* parameter in `server.conf` file.

Return value

Function returns 0 on success and -1 on error.

4.3.4.1.7. query_close function

Close query and free internal data structures.

```
void (SQL_DRV_API *query_close)(SQL_driver *driver);
```

Parameters

driver	[in] Pointer to SQL driver.

Remarks

Server calls this function when all SQL query result are fetched and it needs to free SQL query results stored in DB Driver. This function should free unneeded SQL query results stored in DB Driver.

4.3.4.1.8. get_object function

Create instance of DB driver.

```
SQL_driver *SQL_DRV_API get_object(SQL_driver_config *drv_config);
```

Parameters

driver	[in] Pointer to configuration structure, which keeps lib handle and pointer to logger function.

Remarks

Server calls this function when it needs to initialize generic DB driver with specified DB driver. This function should return generic SQL driver initialized with specific SQL driver.

Return value

Pointer to SQL driver.

4.3.4.1.9. on_delete function

Delete instance of DB driver.

```
void (SQL_DRV_API *on_delete)(SQL_driver *driver);
```

Parameters

driver	[in] Pointer to SQL driver.

Remarks

Server calls this function when it needs to finalize DB driver. Actually this function should do two things: store handle of the actual SQL driver library into SQL driver structure (so it can be accessed by higher-level code), to free it and free the context of the actual SQL driver.

4.3.4.2. Node DB module

This chapter describes functions that need to be implemented in node DB module and functions that is used by node DB module to work with node.

4.3.4.2.1. Functions to work with node

Node provide functions to node DB module to operate on internal node data structures or to execute specific tasks. DB module gets pointers to these internal functions. These pointers to functions are defined in `ext_f.h` file. This is the list of pointers to internal node functions

used by node DB module:

Functions

nget_templates_count	This function returns templates count stored on node.
ndo_match	This function does actual matching job.
nset_current_progress	This function reports progress of matching task.
nresults_add	This function stores matching results to internal nodes data structure.
nget_kill_matching_task	This function checks if matching task received termination command.
nmatching_add_template	This function store fingerprint template received from cluster to node internal data structure.
nmatching_remove_template	This function removes template from node internal data structure.
nresults_free	This function free matching results stored in internal node data structure.
nlprintf	This function is logger function.

4.3.4.2.1.1. nget_templates_count function

This function returns templates count stored on node.

```
int (* nget_templates_count) (void);
```

Return value

Return template count stored on node.

4.3.4.2.1.2. ndo_match function

This function does actual matching job.

```
int (* ndo_match) (unsigned long int t_idx, HNMatcher matcher);
```

Remarks

This function is called when node DB module want to match received fingerprint template

with templates stored in node.

Return value

This function returns similarity on success and -1 on error.

4.3.4.2.1.3. nset_current_progress function

This function reports progress of matching task.

```
void (* nset_current_progress) (int match_count);
```

Remarks

This function is called for node DB module to report matching task progress to clusters server.

4.3.4.2.1.4. nresults_add function

This function stores matching results to internal nodes data structure.

```
void (* nresults_add) (int similarity, char *id, int id_len);
```

Remarks

This function is called for node DB module to store matching results with additional info queried from node database.

4.3.4.2.1.5. nget_kill_matching_task function

This function checks if matching task received termination command.

```
int* (* nget_kill_matching_task) (void);
```

Remarks

This function is called for node DB module to check if node received command to suspend matching task.

Return value

This function should return pointer to kill_matching_task flag. If flag is 1 free results and set kill_matching_flag to 0.

4.3.4.2.1.6. nmatching_add_template function

This function store fingerprint template received from cluster to node internal data structure.

```
int (* nmatching_add_template) (void *template, int size);
```

Remarks

This function is called for node DB module to store templates, received executing db_insert function, on node internal data structures.

Return value

This function should template position in internal node data structure on success and -1 on error.

4.3.4.2.1.7. nmatching_remove_template function

This function removes template from node internal data structure.

```
int (* nmatching_remove_template) (int id);
```

Remarks

This function is called for node DB module to remove templates from node internal data structure.

Return value

Function returns 0 on success and -1 on error.

4.3.4.2.1.8. nresults_free function

This function free matching results stored in internal node data structure.

```
void (* nresults_free) (void);
```

Remarks

This function is called for node DB module to free results stored executing matching task, when this task have been terminated.

4.3.4.2.1.9. nlprintf function

This function is logger function.

```
int (* nlprintf) (const char *format, ...);
```

Remarks

This function is called for node DB module to print log messages.

4.3.4.2.2. Node DB module functions to be implemented

This chapter provides functions that need to be implemented in node DB module. This is the list of these functions:

Functions

db_create	This function creates table on node database.
db_delete	This function deletes database record and template specified by node_dbdelete packet from node database.
db_finalize	This function close DB handler.
db_initialize	This function initialize node database.
db_insert	This function inserts information specified in node_dbinsert packet to node database.
db_matching_loop	This function match received template with templates stored in node internal data structure.
db_setup_node_funcs	This function initializes internal node functions needed by node database module. Also this function initializes node database configuration.

4.3.4.2.2.1. db_create function

This function creates table on node database.

```
int db_create(node_dbcreate *packet)
```

Remarks

This function is called when node receive task form server to create table on nodes database. Table parameters are specified in node_dbcreate packet.

Return value

This function should return 0 on success and -1 on error.

4.3.4.2.2.2. db_delete function

This function deletes database record and template specified by node_dbdelete packet from node database.

```
int db_delete(node_dbdelete *packet)
```

Remarks

Copyright © 2006 Neurotechnologija

This function is called when node receive task from server to delete specific record from node database.

Return value

This function should return 0 on success and -1 on error.

4.3.4.2.2.3. db_finalize function

This function close DB handler.

```
void db_finalize(int *retr);
```

Remarks

This function close DB handler and should set retr to 1. This function is called when node is exiting.

4.3.4.2.2.4. db_initialize function

This function initialize node database.

```
int db_initialize(void);
```

Remarks

This function initialize node database. This function is called when node is starting up.

Return value

This function should return 0 on success and -1 on error.

4.3.4.2.2.5. db_insert function

This function inserts information specified in node_dbinsert packet to node database.

```
int db_insert(node_dbinsert *packet);
```

Remarks

This function is called when node receive task from server to insert received information to node database and store received template in node internal data structure.

Return value

This function should return 0 on success and -1 on error.

4.3.4.2.2.6. db_matching_loop function

This function match received template with templates stored in node internal data structure.

```
int db_matching_loop(char *query, HNMatcher matcher, node_match *task);
```

Remarks

This function receive information from node database specified by SQL query passed to this function. It does matching work by comparing received template with templates stored on node. Templates information is stored in node database and after successful match is returned to server with matching results. This function is called when node receive task from server to match received template with templates store on node.

Return value

This function should return 0 on success and -1 on error.

4.3.4.2.2.7. db_setup_node_funcs function

This function initializes internal node functions needed by node database module. Also this function initializes node database configuration.

```
void db_setup_node_funcs(void *functioned_for_module, void *config_for_module);
```

4.4. Client Components

4.4.1. Template Management

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

- Creating MegaMatcher templates for matcher in Section 4.4.1.1, “Packing NFRecord to NTemplate”.
- Converting VeriFinger/FingerCell templates to MegaMatcher templates in Section 4.4.1.3, “Converting Packed NFRecord from Version 1.0 Format to Current (Version 2.0) Format” and Section 4.4.1.4, “Packing Version 1.0 Format NFRecord to NTemplate”.
- Converting MegaMatcher templates to VeriFinger/FingerCell templates in Section 4.4.1.5, “Converting Packed NFRecord from Current (Version 2.0) Format to Version 1.0 Format” and Section 4.4.1.6, “Retrieving Packed NFRecord in Version 1.0 Format from Packed NTemplate”.

4.4.1.1. Packing NFRecord to NTemplate

High-Level Example (Simpler)

```
[C]

#include "NTemplate.h"
```

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

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

```
[C#]

using System;
using Neurotec.Biometrics;

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

```
                // ...
                // Do something with packed NTemplate
                // ...
        }
}
```

Low-Level Example (Faster and Less Memory Consuming)

```
[C]

#include "NTemplate.h"

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

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

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateManagement
```

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

4.4.1.2. Retrieving Packed NFRecord from Packed NTemplate

High-Level Example (Simpler)

```
[C]

#include "NTemplate.h"

void PackedNFRecordFromPackedNTemplate1(
        NSizeType nTemplateSize, // size of packed NTemplate
        const void * nTemplate // pointer to memory block
                // that contains packed NTemplate
)
{
        HNTemplate hNTemplate; // handle to NTemplate object
        HNFTemplate hNFTemplate; // handle to NFTemplate object
        HNFRecord hNFRecord; // handle to NFRecord object
        NSizeType nfRecordSize; // size of packed NFRecord
        void * nfRecord; // pointer to memory block
                // that contains packed NFRecord

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

```
        NFree(nfRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

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

Low-Level Example (Faster and Less Memory Consuming)

```
[C]

#include "NTemplate.h"

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

        // Unpack packed NFTemplate from packed NTemplate
        NTemplateUnpack(nTemplate, nTemplateSize, NULL, NULL, NULL, NULL,
                &nfTemplate, &nfTemplateSize);
        // Unpack record count from packed NFTemplate
```

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

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

```
[C]

#include "NFRecordV1.h"

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

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

```
[C#]

using System;
using Neurotec.Biometrics;

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

4.4.1.4. Packing Version 1.0 Format NFRecord to NTemplate

To pack version 1.0 NFRecord to NTemplate it has first be converted to current (version 2.0) format. See Section 4.4.1.3, “Converting Packed NFRecord from Version 1.0 Format to Current (Version 2.0) Format” and Section 4.4.1.1, “Packing NFRecord to NTemplate”.

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

```
[C]

#include "NFRecordV1.h"

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

        // Create NFRecord object from packed NFRecord
        NFRecordCreateFromMemory(nfRecord, nfRecordSize, 0, NULL, &hNFRecord);
        // Calculate packed size of NFRecord object in version 1.0 format
        NFRecordGetSizeV1(hNFRecord, flags, &nfRecordV1Size);
        // Allocate memory for packed NFRecord in version 1.0 format
        nfRecordV1 = NAlloc(nfRecordV1Size);
```

```
        // Pack NFRecord object in version 1.0 format
        NFRecordSaveToMemoryV1(hNFRecord, nfRecordV1, nfRecordV1Size,
                flags, &nfRecordSize);
        // Delete NFRecord object
        NFRecordFree(hNFRecord);
        // ...
        // Do something with packed NFRecord in version 1.0 format
        // ...
        // Free memory allocated for packed NFRecord in version 1.0 format
        NFree(nfRecordV1);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateManagement
{
        public void PackedNFRecordToPackedNFRecordV1(
                byte[] packedNFRecord // packed NFRecord
        )
        {
                uint flags = 0; // flags that control packing in version 1.0 format;
                        // specify NFRecord.FlagSkipGs
                        // | NFRecord.FlagSaveBlockedOrients
                        // instead of 0 for VeriFinger 4.1 / FingerCell 1.1
                // Create NFRecord object from packed NFRecord
                NFRecord nfRecord = new NFRecord(packedNFRecord);
                // Pack NFRecord object in version 1.0 format
                byte[] packedNFRecordV1 = nfRecord.SaveV1(flags);
                // ...
                // Do something with packed NFRecord in version 1.0 format
                // ...
        }
}
```

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

To retrieve version 1.0 NFRecord from NTemplate, NFRecord in current (version 2.0) format has first to be retrieved from NTemplate and then converted to version 1.0 format. See Section 4.4.1.2, “Retrieving Packed NFRecord from Packed NTemplate” and Section 4.4.1.5, “Converting Packed NFRecord from Current (Version 2.0) Format to Version 1.0 Format”.

4.5. Additional Components

4.5.1. Template Conversion

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

- Converting MegaMatcher templates to ANSI/INCITS 378-2004 standard templates in Section 4.5.1.4, “Converting NTemplate to FMRecord” and Section 4.5.1.1, “Converting NFRecord to FMRecord”.
- Converting VeriFinger/FingerCell templates to ANSI/INCITS 378-2004 standard templates in Section 4.5.1.2, “Converting Version 1.0 Format NFRecord to FMRecord”.
- Converting two-finger VeriFinger/FingerCell templates to ANSI/INCITS 378-2004 standard templates in Section 4.5.1.3, “Converting Two Version 1.0 Format NFRecords to FMRecord”.
- Converting ANSI/INCITS 378-2004 standard templates to MegaMatcher templates in Section 4.5.1.7, “Converting FMRecord to NTemplate” and Section 4.5.1.5, “Converting FMRecord to NFRecord”.
- Converting ANSI/INCITS 378-2004 standard templates to VeriFinger/FingerCell templates in Section 4.5.1.5, “Converting FMRecord to NFRecord”.
- Converting two-finger ANSI/INCITS 378-2004 standard templates to VeriFinger/FingerCell templates in Section 4.5.1.6, “Converting Two-Finger FMRecord to NFRecords in Version 1.0 Format”.
- Converting MegaMatcher templates to ANSI/NIST-ITL 1-2000 standard files in Section 4.5.1.10, “Converting NTemplate to ANTemplate” and Section 4.5.1.8, “Converting NFRecord to ANTemplate”.
- Converting VeriFinger/FingerCell templates to ANSI/NIST-ITL 1-2000 standard files in Section 4.5.1.9, “Converting Version 1.0 Format NFRecord to ANTemplate”.
- Converting ANSI/NIST-ITL 1-2000 standard files to MegaMatcher templates in Section 4.5.1.12, “Converting ANTemplate to NTemplate” and Section 4.5.1.11, “Converting ANTemplate to NFRecord”.
- Converting ANSI/NIST-ITL 1-2000 standard files to VeriFinger/FingerCell templates in Section 4.5.1.11, “Converting ANTemplate to NFRecord”.

4.5.1.1. Converting NFRecord to FMRecord

```
[C]

#include "FMRecord.h"

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

        // Create NFRecord object from packed NFRecord
        NFRecordCreateFromMemory(nfRecord, nfRecordSize, 0, NULL, &hNFRecord);
        // Create FMRecord object from NFRecord object
        FMRecordCreateFromNFRecord(hNFRecord, 0, &hFMRecord);
        // Delete NFRecord object
        NFRecordFree(hNFRecord);
        // Calculate size of FMRecord object to store in memory
        FMRecordGetSize(hFMRecord, 0, &fmRecordSize);
```

```
        // - or -
        //FMRecordGetSize(hFMRecord, FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Allocate memory for FMRecord
        fmRecord = NAlloc(fmRecordSize);
        // Store FMRecord object in memory
        FMRecordSaveToMemory(hFMRecord, fmRecord, fmRecordSize, 0, &fmRecordSize);
        // - or -
        //FMRecordSaveToMemory(hFMRecord, fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // ...
        // Do something with FMRecord
        // ...
        // Free memory allocated for FMRecord
        NFree(fmRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

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

Remarks

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

.NET) flag should be used.

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

4.5.1.2. Converting Version 1.0 Format NFRecord to FMRecord

Converting version 1.0 format NFRecord to FMRecord is identical to converting NFRecord to FMRecord. See Section 4.5.1.1, “Converting NFRecord to FMRecord”.

4.5.1.3. Converting Two Version 1.0 Format NFRecords to FMRecord

```
[C]

#include "FMRecord.h"

void PackedNFRecordV1ToStoredFMRecordTwoFinger(
        NSizeType nfRecordSize1, // size of first packed NFRecord
        const void * nfRecord1, // pointer to memory block
                // that contains first packed NFRecord
        NSizeType nfRecordSize2, // size of second packed NFRecord
        const void * nfRecord2 // pointer to memory block
                // that contains second packed NFRecord
)
{
        HNFTemplate hNFTemplate; // handle to NFTemplate object
        HNFRecord hNFRecord1; // handle to first NFRecord object
        HNFRecord hNFRecord2; // handle to second NFRecord object
        HFMRecord hFMRecord; // handle to FMRecord object
        NSizeType fmRecordSize; // size of FMRecord
        void * fmRecord; // pointer to memory block that contains FMRecord

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

```
        // FMRFV_USE_NEUROTEC_FIELDS, &fmRecordSize);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // ...
        // Do something with FMRecord
        // ...
        // Free memory allocated for FMRecord
        NFree(fmRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

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

Remarks

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

If CBEFF Product ID is changed then additional Neurotechnologija fields are not created by

default. To force their creation use commented lines(s).

4.5.1.4. Converting NTemplate to FMRecord

```
[C]

#include "FMRecord.h"

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

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

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
```

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

Remarks

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

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

4.5.1.5. Converting FMRecord to NFRecord

```
[C]

#include "FMRecord.h"

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

```
        // Create FMRecord object from FMRecord stored in memory
        FMRecordCreateFromMemory(fmRecord, fmRecordSize, 0, NULL, &hFMRecord);
        // - or -
        //FMRecordCreateFromMemory(fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, NULL, &hFMRecord);
        // Retrieve first FmrFingerView object from FMRecord object
        FMRecordGetFingerView(hFMRecord, 0, &hFmrFingerView);
        // Convert FmrFingerView object to NFRecord object
        FmrFingerViewToNFRecord(hFmrFingerView, 0, &hNFRecord);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // Calculate packed size of NFRecord object
        NFRecordGetSize(hNFRecord, 0, &nfRecordSize);
        // Use the following line instead of previous one
        // if it is needed to pack NFRecord in version 1.0 format
        //NFRecordGetSizeV1(hNFRecord, 0, &nfRecordSize);
        // Allocate memory for packed NFRecord
        nfRecord = NAlloc(nfRecordSize);
        // Pack NFRecord object
        NFRecordSaveToMemory(hNFRecord, nfRecord, nfRecordSize, 0, &nfRecordSize);
        // Use the following line instead of previous one
        // if it is needed to pack NFRecord in version 1.0 format
        //NFRecordSaveToMemoryV1(hNFRecord, nfRecord, nfRecordSize, 0,
        // &nfRecordSize);
        // Delete NFRecord object
        NFRecordFree(hNFRecord);
        // ...
        // Do something with packed NFRecord
        // ...
        // Free memory allocated for packed NFRecord
        NFree(nfRecord);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void StoredFMRecordToPackedNFRecord(
                byte[] storedFMRecord // FMRecord stored in memory
        )
        {
                // Create FMRecord object from FMRecord stored in memory
                FMRecord fmRecord = new FMRecord(storedFMRecord);
                // - or -
                //FMRecord fmRecord = new FMRecord(storedFMRecord,
                // FmrFingerView.UseNeurotecFields);
                // Retrieve first FmrFingerView object from FMRecord object
                FmrFingerView fmrFingerView = fmRecord.FingerViews[0];
                // Convert FmrFingerView object to NFRecord object
                NFRecord nfRecord = fmrFingerView.ToNFRecord();
                // Pack NFRecord object
                byte[] packedNFRecord = nfRecord.Save();
```

```
                // Use the following line instead of previous one
                // if it is needed to pack NFRecord in version 1.0 format
                //byte[] packedNFRecord = nfRecord.SaveV1();
                // ...
                // Do something with packed NFRecord
                // ...
        }
}
```

Remarks

Commented line has to be used if FMRecord was converted from NFRecord and Product Owner part of CBEFF Product ID is set to value different from Neurotechnologija CBEFF Format Owner value and minimal information loss is desired. See also Section 4.5.1.1, “Converting NFRecord to FMRecord”.

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

```
[C]

#include "FMRecord.h"

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

        // Create FMRecord object from FMRecord stored in memory
        FMRecordCreateFromMemory(fmRecord, fmRecordSize, 0, NULL, &hFMRecord);
        // - or -
        //FMRecordCreateFromMemory(fmRecord, fmRecordSize,
        // FMRFV_USE_NEUROTEC_FIELDS, NULL, &hFMRecord);
        // Convert FMRecord object to NFTemplate object
        FMRecordToNFTemplate(hFMRecord, 0, &hNFTemplate);
        // Delete FMRecord object
        FMRecordFree(hFMRecord);
        // Retreive first and second NFRecord objects from NFTemplate
        NFTemplateGetRecord(hNFTemplate, 0, &hNFRecord1);
        NFTemplateGetRecord(hNFTemplate, 1, &hNFRecord2);
        // Calculate packed sizes of first and second NFRecord objects
        // in v 1.0 format
```

Copyright © 2006 Neurotechnologija

```
        NFRecordGetSizeV1(hNFRecord1, 0, &nfRecordSize1);
        NFRecordGetSizeV1(hNFRecord2, 0, &nfRecordSize2);
        // Allocate memory for first and second packed NFRecords
        nfRecord1 = NAlloc(nfRecordSize1);
        nfRecord2 = NAlloc(nfRecordSize2);
        // Pack first and second NFRecord objects in v 1.0 format
        NFRecordSaveToMemoryV1(hNFRecord1, nfRecord1, nfRecordSize1,
                0, &nfRecordSize1);
        NFRecordSaveToMemoryV1(hNFRecord2, nfRecord2, nfRecordSize2,
                0, &nfRecordSize2);
        // Delete NFTemplate object
        NFTemplateFree(hNFTemplate);
        // ...
        // Do something with packed NFRecords
        // ...
        // Free memory allocated for first and second packed NFRecords
        NFree(nfRecord1);
        NFree(nfRecord2);
}
```

```
[C#]

using System;
using Neurotec.Biometrics;

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

Remarks

Commented line has to be used if FMRecord was converted from NFRecord and Product Owner part of CBEFF Product ID is set to value different from Neurotechnologija CBEFF Format Owner value and minimal information loss is desired. See also Section 4.5.1.1, “Converting NFRecord to FMRecord”.

4.5.1.7. Converting FMRecord to NTemplate

```
[C]

#include "FMRecord.h"

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

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

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
```

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

Remarks

Commented line has to be used if FMRecord was converted from NTemplate and Product Owner part of CBEFF Product ID is set to value different from Neurotechnologija CBEFF Format Owner value and minimal information loss is desired. See also Section 4.5.1.4, “Converting NTemplate to FMRecord”.

4.5.1.8. Converting NFRecord to ANTemplate

```
[C]

#include "ANTemplate.h"

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

        // Create NFRecord object from packed NFRecord
        NFRecordCreateFromMemory(nfRecord, nfRecordSize, 0, NULL, &hNFRecord);
```

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

```
[C#]

using System;
using Neurotec.Biometrics;

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

4.5.1.9. Converting Version 1.0 Format NFRecord to ANTemplate

Converting version 1.0 format NFRecord to ANTemplate is identical to converting NFRecord to ANTemplate. See Section 4.5.1.8, “Converting NFRecord to ANTemplate”.

4.5.1.10. Converting NTemplate to ANTemplate

```
[C]

#include "ANTemplate.h"

void PackedNTemplateToANTemplateFile(
        NSizeType nTemplateSize, // size of packed NTemplate
        const void * nTemplate, // pointer to memory block
                // that contains packed NTemplate
        const NAChar * szTot, // pointer to string that specifies
```

```
                // type of transaction
        const NAChar * szDai, // pointer to string that specifies
                // destination agency identifier
        const NAChar * szOri, // pointer to string that specifies
                // originating agency identifier
        const NAChar * szTcn, // pointer to string that specifies
                // transaction control number
        const NChar * szFileName // pointer to string that specifies
                // filename to store ANTemplate
)
{
        HNTemplate hNTemplate; // handle to NTemplate object
        HANTemplate hANTemplate; // handle to ANTemplate object

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

```
[C#]

using System;
using Neurotec.Biometrics;

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

4.5.1.11. Converting ANTemplate to NFRecord

```
[C]

#include "ANTemplate.h"

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

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

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
```

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

4.5.1.12. Converting ANTemplate to NTemplate

```
[C]

#include "ANTemplate.h"

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

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

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

```
[C#]

using System;
using Neurotec.Biometrics;

public class UsingTemplateConversion
{
        public void ANTemplateFileToPackedNTemplate(
                string fileName // name of file were ANTemplate is stored
        )
        {
                // Create ANTemplate object from file
                ANTemplate anTemplate = new ANTemplate(fileName);
                // Convert ANTemplate object to NTemplate object
                NTemplate nTemplate = anTemplate.ToNTemplate();
                // Pack NTemplate object
                byte[] packedNTemplate = nTemplate.Save();
                // ...
                // Do something with packed NTemplate
                // ...
        }
}
```

4.6. Translation of User Interface

To read user interface messages in native language, the messages must be translated. This section describes how it could be done.

These actions must be performed to translate messages:

- download gettext 0.13+ and libiconv-1.9+ (woe32 versions) from any of gnu mirror [6]. Example:

  ```
  ftp://ftp.funet.fi/pub/gnu/prep/gettext/gettext-runtime-0.13.1.bin.woe32.zip
  ftp://ftp.funet.fi/pub/gnu/prep/gettext/gettext-tools-0.13.1.bin.woe32.zip
  ftp://ftp.funet.fi/pub/gnu/prep/libiconv/libiconv-1.9.1.bin.woe32.zip
  ```

- extract downloaded archives to folder. Value of PATH environment variable must contain the folder, where archives where extracted, plus `\bin`. For example, if files where extracted to `C:\gettext\` folder, then PATH could be set this way:

  ```
  set PATH=%PATH%;C:\gettext\bin
  ```

- copy `<nls_domain>.pot` to `<nls_domain>.po` (optionally you can use `msginit' tool from earlier downloaded `gettext' package, which fills some header fields).

[6] http://www.gnu.org/order/ftp.html

`<nls_domain>` is a file, which messages need to be translated.

- Modify `<nls_domain>.po` file:
  - change the `CHARSET' in the header line: "Content-Type: text/plain; charset=CHARSET\n" For example: for Afrikaans, Albanian, Basque, Catalan, Dutch, English, Estonian, Faroese, Finnish, French, Galician, German, Greenlandic, Icelandic, Indonesian, Irish, Italian, Malay, Norwegian, Portuguese, Spanish, Swedish:

```
"Content-Type: text/plain; charset=ISO-8859-1\n"
```

    List of character encodings can be found in gettext manual [7].
  - translate messages. For example:

```
msgid "start server"
msgstr "<translated text here>"
```

    Note
    Not all messages need to be translated.
    Note
    You can edit the file with any text editor or use some specialized tool (like poEdit [8])
- compile `<nls_domain>.po` to `<nls_domain>.mo`:

```
msgfmt --check-format --output-file=<nls_domain>.mo  <nls_domain>.po
```

- copy `<nls_domain>.mo` to folder `locales\<locale>\LC_MESSAGES`. The folder must be created in each working directory of file, which messages need to be translated. To view nls_domain name in use, file, which messages need to be translated, must be executed with `-h` option. For example:

```
Server.exe -h
```

  Result of example:

```
Usage: server.exe [params]
(revision: 4594, nls_domain: server)
```

To use the translation these action must be performed: set variable LANG to value of <locale>(used in name of created folder `locales\<locale>\LC_MESSAGES`) in system environment variables. Execute `<file>` file, for which messages were translated.

[7] http://www.gnu.org/software/gettext/manual/html_node/gettext_35.html#SEC35
[8] http://www.poedit.org/

Chapter 5. Samples

5.1. Oracle and MySQL database tool for Linux OS

This tool is intended for template extraction from images that where retrieved from the file(TIFF, WSQ), and template insertion into Oracle or MySQL database. Precompiled version supports Oracle database. If MySQL database is needed, tool need to be compiled from source. When compiling set MYSQL=yes parameter, for tool to use MySQL module. MegaMatcher NFExtractor module is used for features extraction. The features are packed in NTemplate template. These packed templates are stored in Oracle database.

All functionality of the tool can be achieved by Command Line Interface:

The tool is controlled by these options:

Option	Description
-a	Address of the database server
-u	User name of the database
-p	User password of the database
-d	Name of the database
-t	Table name of the database

Note

If Oracle database is used, address of database should be in following format: / /address_of_oracle_server/Database_name

Input format:

<finger nr> <scan nr> <user id> <wsq|tiff> <dpi> <file name>

Option	Description
<finger nr>	Position of finger
<scan nr>	Number of scanning
<user id>	Person's identifier
<wsq|tiff>	Fingerprint image format: wsq or tiff
<dpi>	Resolution of scanned fingerprint image
<file name>	Filename of scanned fingerprint image

Usage examples:

Extracting the features from image.

```
echo 5 3 John_Peterson wsq 500 JohnPeterson_5_3.wsq | \
./fngr2sql -a <server_ip> -u root -p '' -d cluster -t maintable
```

5.1.1. Files for Linux OS

Required files are:

- ./lib/linux_x86/libNFExtractor.so
- ./lib/linux_x86/libNFTemplate.so
- ./lib/linux_x86/libNTemplate.so
- ./lib/linux_x86/libNCore.so
- ./lib/linux_x86/libNImagesPro.so
- ./lib/linux_x86/libNNFRecord.so
- ./bin/linux_x86/fngr2sql

Folder with source code is `./samples/linux/fngr2sql`

5.2. Sample of Cluster Administrating tool

The sample has its own Command Line Interface. There are several command line options that show how administrating could be done.

The administrating tool is started by:

- for Windows OS: executing `admin.exe` file
- for Linux OS: running the following command `./admin` from folder, where administrating tool file was copied.

There are command line options for controlling the administrating tool.

The tool is controlled by these options:

Option	Value	Description
-s	<server_address>	This option is mandatory. Host or IP address for communicating the Server must be provided.
-p	<server_port>	This option is mandatory. The default value is 24932. Cluster Server port for communication must be

Option	Value	Description
		provided.
-c	<command>	This option is mandatory. The command that should be performed by admin tool must be specified. One of this command could be selected: *start*, *kill*, *stop*, *dbupdate*, *algupdate*, *dbchanged*, *info* . Commands *stop*, *kill*, *dbchanged* and *info* require specifying the *-a* option
-a	<server, node id, row id or requested information type >	This options is mandatory when command options *-c stop* or *-c kill* where specified, as it defines whether the command should be applied to the Server, or to the Node. The exact node is specified by its ID. Command *-c dbchanged* is used to notify server about database changes. -a option defines row id changed in database. Command *info* is used to collect information about cluster. -a option define information type. When -a option is used cluster can provide information about: nodes, running tasks, completed tasks results.
-o	<complete or short information >	This option is mandatory when command *info* and option *tasks*were specified. This option defines long or short information about running task listing.

Source code of tools sample is located:

- `./samples/linux/admin` for Linux OS
- `./samples/Windows/admin` for Windows OS

Binaries of administrating tools sample are located:

- `./bin/linux_x86/admin` for Linux OS
- `./bin/Win32_x86/admin.exe` for Windows OS

5.3. Client sample for Linux OS

The primary purpose of this tool is to show how to use the Cluster, or how to write custom client software. This tool is capable to extract templates from images, form a task batch and send the batch to the Server. Then the Server informs the Client about the task completion; the Client receives results and deletes results from the Server.

5.3.1. Using Command Line Interface for controlling the Cli-

ent application

The functionality of this tool is controlled using Command Line Interface.

5.3.1.1. Client command line options

The Client is controlled by these options:

Option	Value	Description
-s	<server_address>	This option is mandatory. Host or IP address for communicating the Server must be provided.
-p	<server_port>	This option is mandatory. The default value is 24932. Cluster Server port for communication must be provided.
-d	<dpi>	The default value is 250. The image resolution could be specified by this option. If a new task batch is going to be sent and the image resolution isn't 250 DPI, a correct value must be specified.
-a	<tiff_image>	Either this option or -r option must be provided. This option specifies the image for extraction of template and task creation. This sample client tool supports only TIFF image format due simplicity of the Client tool.
-f	-	This option specifies that only the first result of the new task should be returned.
-n	<scanner_id>	This option specifies that fingerprint image should be taken from the scanner. The scanner type is specified by scanner_id. • 0 - Biometrika FX2000 • 1 - Shimizu Tacoma • 2 - Fujitsu MBF200 • 3 - Startek FM200 • 4 - Cross Match Verifier 300-USB
-r	<task_id>	Either this option or -a option must be provided. This option specifies identifier of task (task_id). The tool tries to get the task progress information from the Server. If the task is complete, the tool displays the results and then removes the results from the Server.

Copyright © 2006 Neurotechnologija

5.3.1.2. Submitting the tasks

A task could be submitted by either command line options of client tool, or .NET component.

5.3.1.3. Retrieving the results

Results are stored in the Server. These results can be retrieved by either command line options of client tool, or .NET component.

5.3.2. Files

Required files are:

- ./lib/linux_x86/libNFExtractor.so
- ./lib/linux_x86/libNFTemplate.so
- ./lib/linux_x86/libNTemplate.so
- ./lib/linux_x86/libNCore.so
- ./lib/linux_x86/libNImagesPro.so
- ./lib/linux_x86/libNFRecord.so
- ./bin/linux_x86/client

Folder with source code is `./samples/linux/client`

5.4. Client application for .NET

This sample was developed to show how all the Client Side Components work and could be used. The Sample also includes Database tool to store enrolled fingerprints into Database.

Sample applications are provided for Microsoft Visual C#

Source code for application is located in .\samples\dotNET\MatchingClusterClientSample folder. The solution file is MatchingClusterClientSample.sln.

Before running the sample, the working directory in project properties should be set to "`%MEGA_MATCHER_SDK%\bin\%PLATFORM_ID%`", where "`%MEGA_MATCHER_SDK%`" is a root of MegaMatcher SDK installation and "%PLATFORM_ID%" is an appropriate platform (for example "Win32_x86").

A reference to `MySql.Data.dll` should be added into project **MatchingClusterClientSample**.

5.4.1. Description

C# cluster client sample demonstrates how the Client application may receive fingerprints, enroll them to the database and communicate with a MegaMatcher Cluster. *`MatchingClusterClientSample.bat`* should be executed to start the sample application.

After running the application, the Client should be registered by opening the menu **Help** and selecting **Register** option.

5.4.2. Running the demo application

- .NET framework 1.1 , Oracle Data Provider for .NET and MySQL Connector/Net 1.0 should be installed before running demo application.

5.4.3. User interface

5.4.3.1. Main window

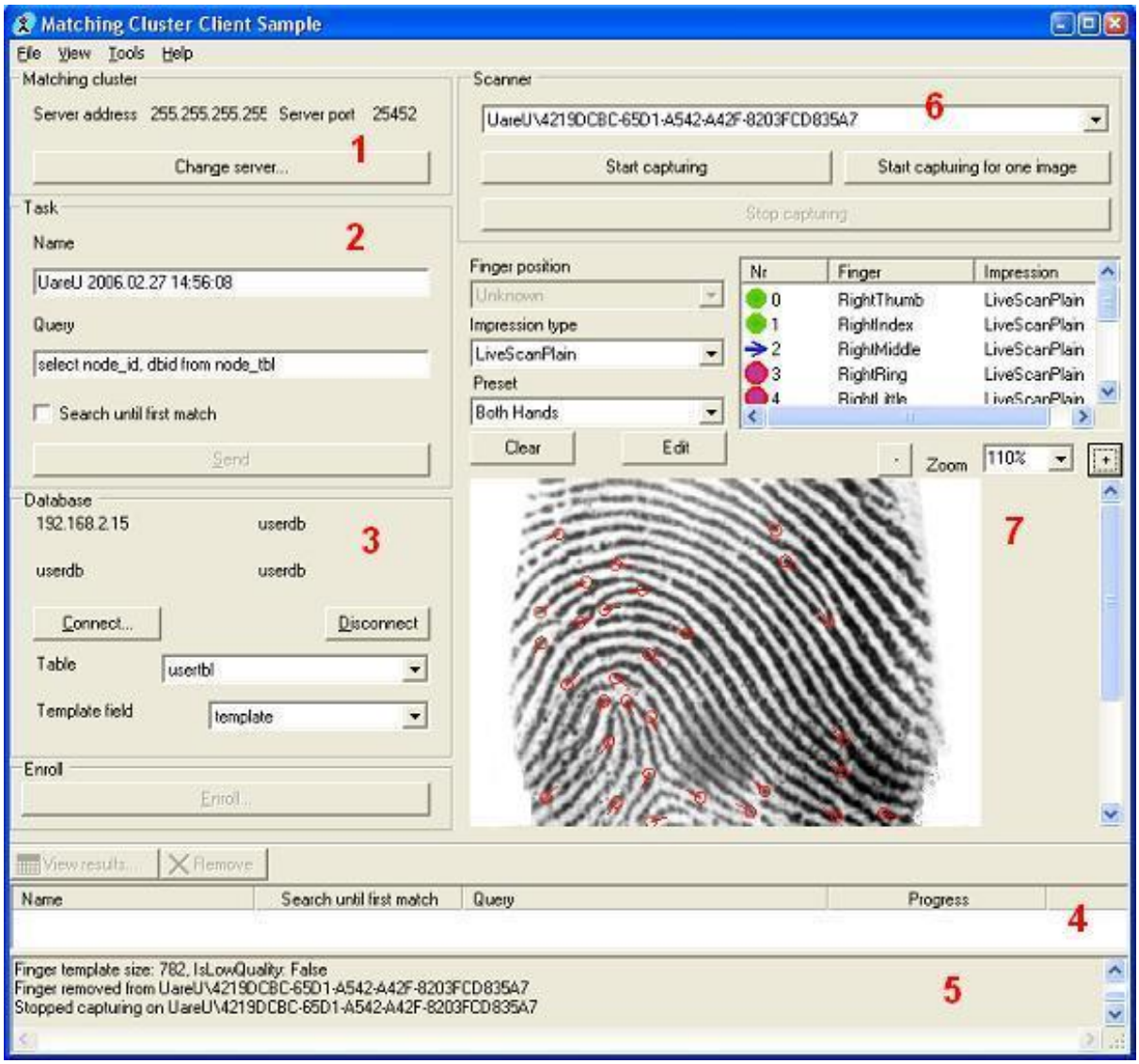

Figure 5.1. Screenshot of Client for .NET sample application

1. `Matching cluster view` - dedicated for showing the server information and changing it.
2. `Task view` - dedicated for sending data to the server. The matching query for nodes (default query: *select node_id, db_id from node_tbl*).
3. `Database view` - dedicated for fingerprint enroll in DB.
4. `Task in progress view` - dedicated for viewing task progress.
5. `Log view` - dedicated for error and additional information.
6. `Scanner view` - dedicated for start, stop capturing from scanner.
7. `Finger print view` - shows fingerprint and binarized or skeletonized image.

5.4.3.2. Main menu

1. `File` - operations with files.

Operation	Description
`Open`	Loads an image from the file
`Save Image`	Saves a fingerprint image to the file
`Save Result Image`	Saves the result (after features extraction) image to the file
`Exit`	Closes the application
`Clear`	clears Finger print view

2. `View` - controls fingerprint view.

Control	Description
`Original`	Shows the original image (if image is scanned or opened from file)
`Show Result Image`	Shows binarized or skeletonized image (if image is scanned or opened from file)
`Show Minutiae`	Shows/hides minutiae

3. `Tools` - enables change parameters.

Tool	Description
`Extractor Parameters`	Enables changing of parameters for Extract-or
`Matcher Parameters`	Enables changing of parameters for different

Tool	Description
	Matchers
`Use VF Extractor`	Application uses VFExtractor if this option is selected

4. `Help` - about C# client sample.

Item	Description
`Register`	Enables registration of this Matching Cluster Client Sample application on server. Requires entering Server name or ip address
`About`	Shows information about this Matching Cluster Client Sample application

5.4.3.3. Buttons

Button	Description
Start capturing	Starts capturing fingerprints for selected scanner. Unhandled exception occurs if this button was clicked more than once for the same scanner: "Error starting capturing on FPScannerMan scanner. Code: -2147217404"
`Stop capturing`	Stops capturing for selected scanner
`Start capturing for one image`	Starts capturing only for one image
`Change Server`	For setting the Matching Server information
`Connect...`	For connecting to the main database
`Disconnect`	For disconnecting from the main database
`Enroll..`	For fingerprint enrollment
`Send`	Sends the template to the Cluster
`Remove`	Removes the task from task list and the server
`Edit`	Starts fingerprint record editor

Copyright © 2006 Neurotechnologija

Button	Description
`Clear`	Clears data for selected preset
`+, -`	Zooms image in and out

5.4.3.4. Features

Feature	Description
`Capture fingerprint`	Fingerprint can be captured either from scanner by clicking button `Start capturing` or from file
`Enroll fingerprint into database`	Fingerprint can be enrolled into database. Before enrollment `Database view` area should be filled up. After processing `Enroll...` button, column name specified as `UidName` in server configuration file should be chosen. Also table column name should be chosen and the value entered. If UidName is as **id** column chosen, then this column is not need to fill(this field is auto_increment). EnrollForm form will insert template to database and send message to server that new entry was enrolled, then nodes are informed of database changes.
`Send template to cluster for identification`	Before sending the fingerprint template to cluster. Cluster information `Server address`, `port`, `query` should be entered and finger position should be selected in ComboBox. If CheckBox `Search until first match` was checked and the finger is found in database, then the result is returned. Otherwise all results are returned from the server (identifier and score).
`Get results of identification from Cluster`	Task progress can be viewed after sending the task. Results from the Server can be viewed by clicking button `View results`

5.4.3.5. Code description

Code	Description
`MainForm`	Class captures fingers, shows minutiae, sends templates to the Cluster and gets results from the Cluster(visualizes results). Objects `scannerMan, nfExtractor, vfExtractor, matchingClusterClient` are created by running C# demo application.
`startCapturingButton_Click`	Starts capturing from the scanner
`scanner_ImageScanned`	Gets scanned image
`OnImage`	Extracts template, makes binarized or skeletonized image
`taskSendButton_Click`	Sends template to cluster

5.4.4. Cluster communication

Component consists of: structure and classes.

Classes

MatchingClusterClient	The class was designed to create, delete and manage tasks (objects of MatchingTask class)
MatchingTask	This class was designed for sending tasks to the Server and receiving the results.

Structures

MatchingTaskParameter	The structure was designed to store parameters of matching algorithm.

5.4.4.1. MatchingClusterClient Class

Constructors

MatchingClusterClient Constructor	Initializes a new instance of the `MatchingClusterClient` class.

Properties

ServerAddress	Gets or sets server address.
ServerPort	Gets or sets server port.
Tasks	Gets the collection of tasks.

Methods

Dispose	Removes task, stops thread.
HasPendingTask	Returns `true` if there are more executing tasks; otherwise, `false`.

5.4.4.1.1. MatchingClusterClient Constructors

Initializes a new instance of the `MatchingClusterClient` class.

```
public MatchingClusterClient(
        ISynchronizeInvoke synInvoke
);
```

Parameters

synInvoke	The object Form(class is using in Windows Form), or null for other cases.

5.4.4.1.2. ServerAddress Property

Gets or sets server address.

```
IPAddress ServerAddress {get; set;}
```

Property value

The server address.

5.4.4.1.3. ServerPort Property

Gets or sets server port.

```
int ServerPort {get; set;}
```

Property value

The server port.

5.4.4.1.4. Tasks Property

Gets the collection of tasks.

```
MatchingClusterClient.TaskCollection Tasks {get;}
```

Property value

The tasks collection.

5.4.4.1.5. Dispose Method

Removes task, stops thread.

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

5.4.4.1.6. HasPendingTask Method

Returns `true` if there are more executing tasks; otherwise, `false`.

```
bool HasPendingTask();
```

Return Values

`true` if MatchingClusterClient has more executing tasks; otherwise, `false`.

5.4.4.2. MatchingTask Class

Constructors

MatchingTask Constructor	Initializes a new instance of the `MatchingTask` class.

Properties

Error	Gets the communication exception.
IsComplete	Gets the task completion status.
IsStarted	Gets the task state.
Name	Gets the task name.

Parameters	Gets the parameters for matcher.
Progress	Gets the task progress.
Query	Gets the query for database.
Results	Gets the results from the server.
SearchUntilFirstMatch	Gets a value indicating matching method.
Template	Gets the packed NTempalte object.

5.4.4.2.1. MatchingTask Constructors

Initializes a new instance of the `MatchingTask` class.

```
MatchingTask(
        string name,
        bool searchUntilFirstMatch,
        byte[] template,
        string query,
        params MatchingTaskParameter[] parameters
);
```

Parameters

name	The task name.
searchUntilFirstMatch	`true` matches till first matching score is greater than threshold. Otherwise, matches with all database.
template	The packed NTemplate object.
query	The query for database.
parameters	The parameters for matcher.

5.4.4.2.2. Error Property

Gets the communication exception.

```
Exception Error {get;}
```

Property value

The communication exception.

5.4.4.2.3. IsComplete Property

Gets the task completion status.

```
bool IsComplete {get;}
```

Property value

`true` if task is completed; otherwise, `false`.

5.4.4.2.4. IsStarted Property

Gets the task state.

```
bool IsStarted {get;}
```

Property value

`true` if task is started; otherwise, `false`.

5.4.4.2.5. Name Property

Gets the task name.

```
string Name {get;}
```

Property value

The task name.

5.4.4.2.6. Parameters Property

Gets the parameters for matcher.

```
MatchingTaskParameter[] Parameters {get;}
```

Property value

The parameters for matcher.

5.4.4.2.7. Progress Property

Gets the task progress.

```
int Progress {get;}
```

Property value

The task progress, the number is from interval [0, 100].

5.4.4.2.8. Query Property

Gets the query for database.

```
string Query {get;}
```

Property value

The query for database.

5.4.4.2.9. Results Property

Gets the results from the server.

```
DataTable Results {get;}
```

Property value

The results from the server. The table with two columns: identifier(id) and similarity(score).

5.4.4.2.10. SearchUntilFirstMatch Property

Gets a value indicating matching method.

```
bool SearchUntilFirstMatch {get;}
```

Property value

`true` matches till first matching score is greater than threshold; otherwise, `false` if matches with all database.

5.4.4.2.11. Template Property

Gets the packed NTempalte object.

```
byte[] Template {get;}
```

Property value

The packed NTempalte object.

5.4.4.3. MatchingTaskParameter Struct

Constructors

MatchingTaskParameter Constructor	Initializes a new instance of the `MatchingTaskParameter` structure.

Properties

ID	Gets parameter identifier.
PartID	Get the part of NMatcher algorithm for which sets the value of parameter.
Value	Gets parameter value.

5.4.4.3.1. MatchingClusterClient Constructors

Initializes a new instance of the `MatchingTaskParameter` structure.

```
MatchingTaskParameter(
        ushort partId,
        ushort id,
        object value
);
```

Parameters

partId	Defines part of NMatcher algorithm for which sets the value of parameter. *See also:* GetParameter.
id	The parameter identifier.
value	The value of parameter.

5.4.4.3.2. ID Property

Gets parameter identifier.

```
ushort ID {get;}
```

Property value

The parameter identifier.

5.4.4.3.3. PartID Property

Get the part of NMatcher algorithm for which sets the value of parameter.

```
ushort PartID {get;}
```

Property value

The part of NMatcher algorithm for which sets the value of parameter.

See Also

GetParameter

5.4.4.3.4. Value Property

Gets parameter value.

```
object Value {get;}
```

Property value

The parameter value.

5.4.5. Fingerprint record editor

Figure 5.2. Screenshot of fingerprint record editor window

This dialog allows to change minutiae data for selected fingerprint. To select minutia, click over it. Selected minutia marked magenta. Neighbours of selected minutia marked orange. When mouse moves over neighbour minutia, ridge count are displayed in tool tip control. Other properties of minutia are displayed right in property grid. Property grid allows to change minutia properties manual.

5.4.5.1. Buttons

Button	Description
Add minutia	Adds minutia to fingerprint record. Added minutia can be moved to right position.
Remove minutia	Removes minutia from fingerprint record.
Recalculate	Recalculates ridge counts, curvatures and G for current fingerprint.
Buttons with straight arrows	Moves selected minutia.
Buttons with broken arrows	Rotates selected minutia.
Buttons with + and -	Zoom in and zoom out.
OK	Closes dialog. Recalculates and saves changes.
Cancel	Closes dialog. Ignores changes made.

5.5. Client application for Visual Basic 6

This sample was developed to show how Client Side Components work and could be used under Visual Basic 6. The Sample also includes Database tool to store enrolled fingerprints into Database. Sample source code is located in .\samples\VB\MatchingClusterClientSample folder. The project file is `MatchingClusterClientSample.vbp`.

5.5.1. Description

Visual Basic 6 cluster client sample demonstrates how the Client application may capture fingerprints, enroll them to the database and communicate with a MegaMatcher Cluster. After running the application, the Client should be registered by opening the menu **Help** and selecting **Register** option.

5.5.2. Running the demo application

Folder/File	
dlls	
ANTemplate.dll	
FMRecord.dll	
FPScannerMan.dll	
FPScannerManCom.dll	
NCore.dll	
NFExtractor.dll	
NFMatcher.dll	
NFRecord.dll	
NFTemplate.dll	
NImagesPro.dll	
NTemplate.dll	
VFExtractor.dll	
	fpsmm
	FPSmmAtmel.dll
	FPSmmBiometrika.dll
	FPSmmCrossMatch.dll
	FPSmmFM200.dll
	FPSmmIdentix.dll
	FPSmmLighTuning.dll
	FPSmmST.dll
	FPSmmTacoma.dll
	FPSmmUareU.dll

Important

Before running the sample, the System variables ("Path") should be set to the folder where files from above table is located. Otherwise, these files must be copied to Windows System

directory. `FPScannerManCom.dll` must be registered before running this sample. Use *`regFPScannerManCom.bat`* to register this dll.

5.5.3. User interface

Refer to C# demo for detailed interface description.

5.5.4. Wrappers

This chapter describes wrappers for NFExtractor, VFExtractor, NFRecord, NFTemplate, NTemplate modules and NImages Pro library.

5.5.4.1. modNFTemplate Module

The wrapper for NFTemplate consists of modNFTemplate class module.

Methods

NFTemplatePack	Packs packed finger records as NFTemplate into the byte array.

5.5.4.1.1. NFTemplatePack Method

Packs packed finger records as NFTemplate into the byte array.

```
Public Sub NFTemplatePack(
        ByVal recordCount As Long,
        ByRef arRecords() As Long,
        ByRef arRecordSizes() As Long,
        ByRef packetRecords() As Byte
)
```

Parameters

`recordCount`	The number of finger records.
`arRecords`	The array that contains addresses of packed finger records.
`arRecordSizes`	The array that contains packed sizes of each finger record.
`packetRecords`	The byte array to store packed NFTemplate.

5.5.4.2. modNImages Module

The wrapper for NImagesPro consists of modNImages, modWSQ modules.

Methods

BmpSaveImageToHBitmap	Saves image to Windows HBITMAP.
NImageCreateFromData	Creates an image with specified pixel format, size, stride and resolution.
NImageCreateFromFile	Creates (loads) an image from file of specified format.
NImageCreateFromImage	Creates an image from specified image with specified pixel format and stride.
NImageSaveToFile	Saves the image to the file of specified format.
NImagesIsRegistered	Checks if NImages Pro library is registered.
NImagesRegister	Registers NImages Pro library.
NImagesUnregister	Unregisters NImages Pro library.
Picture	Makes the IPictureDisp object.

5.5.4.2.1. BmpSaveImageToHBitmap Method

Saves image to Windows HBITMAP.

```
Public Function BmpSaveImageToHBitmap(
        ByVal nImage As Long
) As Long
```

Parameters

nImage	Handle to image.

Return Values

The handle to created Windows HBITMAP.

5.5.4.2.2. NImageCreateFromData Method

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

```
Public Function NImageCreateFromData(
        ByVal pixelFormat As NPixelFormat,
```

```
        ByVal width As Long,
        ByVal height As Long,
        ByVal stride As Long,
        ByVal horzResolution As Single,
        ByVal vertResolution As Single,
        ByVal pixelsStride As Long,
        ByRef pixels() As Byte
) As Long
```

Parameters

pixelFormat	Specifies pixel format of the image.
width	Specifies width of the image.
height	Specifies height of the image.
stride	Specifies stride of the image. Can be zero.
horzResolution	Specifies horizontal resolution in pixels per inch of the image.
vertResolution	Specifies vertical resolution in pixels per inch of the image.
pixelsStride	Specifies stride of pixels to be copied to the image.
pixels	The byte array with the image.

Return Values

The handle to created image.

See Also

NImageCreateFromData | NImageCreateFromFile

5.5.4.2.3. NImageCreateFromFile Method

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

```
Public Function NImageCreateFromFile(
        ByVal fileName As String,
        ByVal imageFormat As Long
) As Long
```

Parameters

fileName	A string that contains the name of the file.
imageFormat	The image format of the file.

Return Values

The address to the handle to created image.

See Also

NImageSaveToFile

5.5.4.2.4. NImageCreateFromImage Method

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

```
Public Function NImageCreateFromImage(
        ByVal pixelFormat As NPixelFormat,
        ByVal stride As Long,
        ByVal nImage As Long
) As Long
```

Parameters

pixelFormat	Specifies pixel format of the image.
stride	Specifies stride of the image. Can be zero.
nImage	Handle to image used as source for the image.

Return Values

The handle to created image.

See Also

Picture ExtractFromImage

5.5.4.2.5. NImageSaveToFile Method

Saves the image to the file of specified format.

```
Public Sub NImageSaveToFile(
        ByVal nImage As Long,
        ByVal fileName As String,
        ByVal imageFormat As Long
)
```

Parameters

nImage	The address to NImage object.
fileName	A string that contains the name of the file.
imageFormat	The image format of the file.

Return Values

If *imageFormat* is 0 image format is selected by file extension of *fileName*.

See Also

NImageCreateFromData | NImageCreateFromFile

5.5.4.2.6. NImagesIsRegistered Method

Checks if NImages Pro library is registered.

```
Public Function NImagesIsRegistered() As Boolean
```

Return Values

TRUE if library is registered, FALSE otherwise.

See Also

NImagesRegister | NImagesUnregister

5.5.4.2.7. NImagesRegister Method

Registers NImages Pro library.

```
Public Sub NImagesRegister(
        ByVal server As String
)
```

Parameters

server	The string that specifies license manager server name or IP address (if registering with LAN License Manager), any string (if registering with HASP) or serial number and registration key separated by '@' (if registering using PC or U.are.U protection).

Remarks

If registering with HASP it is recommended that parameter *server* should be an empty string.

If registering with PC/U.are.U protection parameter *server* should be the string in format "serial_number@registration_key".

When library is no longer needed (possibly during application shutdown) it must be unregistered using NImagesUnregister method.

See Also

NImagesUnregister | NImagesIsRegistered

5.5.4.2.8. NImagesUnregister Method

Unregisters NImages Pro library.

```
Public Sub NImagesUnregister()
```

See Also

NImagesRegister | NImagesIsRegistered

5.5.4.2.9. Picture Method

Makes the IPictureDisp object.

```
Public Function Picture(
        ByVal nImage As Long
) As IPictureDisp
```

Parameters

nImage	The handle to the image.

Return Values

The IPictureDisp object.

See Also

NImageCreateFromImage

5.5.4.3. modNTemplate Module

The wrapper for NFTemplate consists of modNTemplate class module.

Methods

NTemplatePack	Packs packed fingers template as NTemplate into the byte array.

5.5.4.3.1. NTemplatePack Method

Packs packed fingers template as NTemplate into the byte array.

```
Public Sub NTemplatePack(
        ByRef fingersTemplates() As Byte,
        ByRef buffer() As Byte
)
```

Parameters

fingersTemplates	The byte array that contains packed NFTemplate.
buffer	The byte array that contains packed NTemplate.

5.5.4.4. modWSQ Module

To use NImages Pro library you have to register it. See NImagesRegister.

Methods

WsqLoadImageFromFile	Loads image from WSQ file.
WsqSaveImageToFile	Saves image to file in WSQ format with specified bitrate.

5.5.4.4.1. WsqLoadImageFromFile Method

Loads image from WSQ file.

```
Public Function WsqLoadImageFromFile(
        ByVal fileName As String
) As Long
```

Parameters

fileName	A string that contains the name of the file.

Return Values

See Also

WsqSaveImageToFile

5.5.4.4.2. WsqSaveImageToFile Method

Saves image to file in WSQ format with specified bitrate.

```
Public Sub WsqSaveImageToFile(
        ByVal nImage As Long,
        ByVal fileName As String,
        ByVal bitRate As Single
)
```

Parameters

nImage	Handle to image.
fileName	A string that contains the name of the file.
bitRate	Specifies bit rate (compression level).

See Also

WsqLoadImageFromFile

5.5.4.5. NFExractor Class

The wrapper for NFExtractor consists of Visual Basic 6NFExractor class module and modN-FExtractor module.

Constructors

NFExractor Constructor	Initializes a new instance of the `NFExractor` class.

Properties

GetMaxTemplateSize	Calculates maximal size of template.

GetParameter	Gets value of parameter.
Handle	Gets handle of class object.
IsRegistered	Checks if the NFExtractor library is registered.

Methods

CopyParameters	Copies values of parameters from one NFExtractor object to another.
Extract	Extracts features from fingerprint image.
ExtractFromImage	Extracts features from NImage.
Register	Registers the NFExtractor library. Registered library must be unregistered using `Unregister` function.
Reset	Resets all NFExtractor parameters to default values.
SetParameter	Sets parameter value.
Unregister	Unregisters the NFExtractor library.

Events

ErrorOccured	Occurs when the error is occred in class or module.

5.5.4.5.1. NFExtractor Constructor

Initializes a new instance of the `NFExractor` class.

```
Public Sub New()
```

5.5.4.5.2. CopyParameters Method

Copies values of parameters from one NFExtractor object to another.

```
Public Sub CopyParameters(
        ByRef sourceExractor As Long
)
```

Parameters

sourceExractor	Handle to the NFExtractor object.

See Also

SetParameter | GetParameter

5.5.4.5.3. ErrorOccured Event

Occurs when the error is occred in class or module.

```
Public Event ErrorOccured(
        msg As String,
        errorCode As Long
)
```

Parameters

msg	The error message.
errorCode	The error code.

5.5.4.5.4. Extract Method

Extracts features from fingerprint image.

```
Public Sub Extract(
        ByVal width As Integer,
        ByVal height As Integer,
        ByVal stride As Integer,
        ByVal horzResolution As Integer,
        ByVal vertResolution As Integer,
        ByRef pixels() As Byte,
        ByVal position As Long,
        ByVal impressionType As Long,
        ByRef buffer() As Byte,
        ByRef isLowQuality As Boolean
)
```

Parameters

width	Specifies fingerprint image width.
height	Specifies fingerprint image height.
stride	Specifies fingerprint image width, plus the

	width of the alignment bytes.
horzResolution	Specifies horizontal resolution of fingerprint image.
vertResolution	Specifies vertical resolution of fingerprint image.
pixels	The array of fingerprint image.
position	Specifies position.
impressionType	Specifies impression type.
buffer	The buffer to store packed NFRecord.
isLowQuality	TRUE if fingerprint image is low quality, FALSE if quality is high.

See Also

ExtractFromImage

5.5.4.5.5. ExtractFromImage Method

Extracts features from NImage.

```
Public Sub ExtractFromImage(
        ByVal image As Long,
        ByVal position As Long,
        ByVal impressionType As Long,
        ByRef buffer() As Byte,
        ByRef isLowQuality As Boolean
)
```

Parameters

image	Handle to the NFExtractor object.
position	Handle to the NFExtractor object.
impressionType	Specifies impression type.
buffer	The buffer to store packed NFRecord.
isLowQuality	TRUE if fingerprint image is low quality, FALSE if quality is high.

See Also

Extract

5.5.4.5.6. GetMaxTemplateSize Property

Calculates maximal size of template.

```
Public Property Get GetMaxTemplateSize() As Long
```

Property value

The maximal template size.

5.5.4.5.7. GetParameter Property

Gets value of parameter.

```
Public Property Get GetParameter(
        ByVal parameterId As Long
) As Variant
```

Property value

The parameter value.

See Also

SetParameter | CopyParameters

5.5.4.5.8. Handle Property

Gets handle of class object.

```
Public Property Get Handle() As Long
```

Property value

The handle of class object.

5.5.4.5.9. IsRegistered Property

Checks if the NFExtractor library is registered.

```
Public Property Get IsRegistered() As Boolean
```

Property value

TRUE if library is registered, FALSE if library is not registered.

See Also

Register | Unregister

5.5.4.5.10. Register Method

Registers the NFExtractor library. Registered library must be unregistered using `Unregister` function.

```
Public Sub Register(
        ByVal server As String
)
```

Parameters

server	Specifies registration server name.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

5.5.4.5.11. Reset Method

Resets all NFExtractor parameters to default values.

```
Public Sub Reset()
```

See Also

SetParameter | GetParameter | CopyParameters

5.5.4.5.12. SetParameter Method

Sets parameter value.

```
Public Sub SetParameter(
        ByVal parameterId As Long,
        ByVal value As Long
)
```

Parameters

parameterId	Parameter identifier to set.
value	Parameter value to set.

See Also

GetParameter | CopyParameters

5.5.4.5.13. Unregister Method

Unregisters the NFExtractor library.

```
Public Sub Unregister()
```

See Also

Register | IsRegistered

5.5.4.6. NFRecord Class

The wrapper for NFRecord consists of Visual Basic 6 NFRecord class module and modNFRecord module.

Constructors

NFExractor Constructor	Initializes a new instance of the `NFRecord` class.

Properties

RecordGetMinutiaCount	Retrieves the number of minutiae in the NFRecord.

Methods

RecordCreateFromMemory	Unpacks a NFRecord from the byte array.
RecordGetMinutia	Retrieves the minutia at the specified index of the NFRecord.
RecordGetMinutiae	Copies all minutiae of NFRecord to the specified array.

Events

ErrorOccured	Occurs when the error is occred in class or module.

5.5.4.6.1. NFRecord Constructor

Initializes a new instance of the `NFRecord` class.

```
Public Sub New()
```

5.5.4.6.2. ErrorOccured Event

Occurs when the error is occred in class or module.

```
Public Event ErrorOccured(
        msg As String,
        errorCode As Long
)
```

Parameters

msg	The error message.
errorCode	The error code.

5.5.4.6.3. RecordCreateFromMemory Method

Unpacks a NFRecord from the byte array.

```
Public Sub RecordCreateFromMemory(
        ByRef fingerTemplate() As Byte,
        ByVal flags As Integer
)
```

Parameters

fingerTemplate	The byte array that contains packed NFRecord.
flags	Bitwise combination of zero or more flags that controls behavior of the method.

5.5.4.6.4. RecordGetMinutia Method

Retrieves the minutia at the specified index of the NFRecord.

```
Friend Sub RecordGetMinutia(
        ByVal index As Long,
        ByRef minutia As NFMinutia
)
```

Parameters

index	Index of minutia to retrieve.
minutia	NFMinutia structure that receives minutia.

See Also

RecordGetMinutiaCount | RecordGetMinutiae

5.5.4.6.5. RecordGetMinutiaCount Property

Retrieves the number of minutiae in the NFRecord.

```
Public Property Get RecordGetMinutiaCount() As Long
```

Property value

The number of minutiae.

5.5.4.6.6. RecordGetMinutiae Method

Copies all minutiae of NFRecord to the specified array.

```
Friend Sub RecordGetMinutiae(
        ByRef arMinutiae() As NFMinutia
)
```

Parameters

arMinutiae	The array of NFMinutia that receives minutiae.

See Also

RecordGetMinutiaCount

5.5.4.7. VFExractor Class

The wrapper for VFExtractor consists of VFExractor class module and modVFExtractor module.

VFExtractor class has the almost same named methods, properties and events as NFExtractor

Chapter 6. Reference (C/C++)

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

Libraries

ANTemplate	Provides functionality for converting ANSI/NIST-ITL 1-2000 standard files to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.
FMRecord	Provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.
NCore	Provides infrastructure for Neurotechnologija components.
NFExtractor	Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using MegaMatcher algorithm.
NFMatcher	Provides functionality for comparing Neurotechnologija Finger Records using MegaMatcher algorithm.
NFRecord	Provides functionality for packing, unpacking and editing Neurotechnologija Finger Records.
NfsMatcher	Provides functionality for comparing Neurotechnologija Fingers Templates.
NFTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates.
NImages Pro	Provides functionality for loading, saving and converting images in various formats.
NMatcher	Provides functionality for comparing Neurotechnologija Templates.
NTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Tem-

	plates.
VFExtractor	Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using VeriFinger algorithm.
VFMatcher	Provides functionality for comparing Neurotechnologija Finger Records using VeriFinger algorithm.

6.1. ANTemplate Library

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

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

DLL (Windows): `ANTemplate.dll`.

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

Requirements (Windows):

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

Requirements (Linux):

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

Modules

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

6.1.1. ANTemplate Module

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

Header file: `ANTemplate.h`.

Functions

`ANTemplateCreateFromFile`	Loads an ANTemplate from the specified file.
`ANTemplateCreateFromNFRecord`	Creates an ANTemplate from the specified NFRecord.
`ANTemplateCreateFromNFTemplate`	Creates an ANTemplate from the specified NFTemplate.
`ANTemplateCreateFromNTemplate`	Creates an ANTemplate from the specified NTemplate.
`ANTemplateFree`	Deletes the ANTemplate. After the object is deleted the specified handle is no longer valid.
`ANTemplateSaveToFile`	Saves the ANTemplate to the specified file.
`ANTemplateToNFTemplate`	Converts the ANTemplate to a NFTemplate.
`ANTemplateToNTemplate`	Converts the ANTemplate to a NTemplate.

Types

HANTemplate	Handle to an ANTemplate object.

Macros

ANT_MAX_DIMENSION	For internal use.

See Also

ANTemplate Library

6.1.1.1. ANTemplateCreateFromFile Function

Loads an ANTemplate from the specified file.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Created object must be deleted using `ANTemplateFree` function.

See Also

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

6.1.1.2. ANTemplateCreateFromNFRecord Function

Creates an ANTemplate from the specified NFRecord.

```
NResult N_API ANTemplateCreateFromNFRecord(
        HNFRecord hNFRecord,
        const NChar * szTot,
        const NChar * szDai,
        const NChar * szOri,
```

```
        const NChar * szTcn,
        HANTemplate * pHANTemplate
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Created object must be deleted using `ANTemplateFree` function.

See Also

ANTemplate Module | HANTemplate | `ANTemplateFree` | HNFRecord

6.1.1.3. ANTemplateCreateFromNFTemplate Function

Creates an ANTemplate from the specified NFTemplate.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Created object must be deleted using `ANTemplateFree` function.

See Also

ANTemplate Module | HANTemplate | ANTemplateFree | HNFTemplate | ANTemplateToNFTemplate

6.1.1.4. ANTemplateCreateFromNTemplate Function

Creates an ANTemplate from the specified NTemplate.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Copyright © 2006 Neurotechnologija

Remarks

Created object must be deleted using `ANTemplateFree` function.

See Also

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

6.1.1.5. ANTemplateFree Function

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

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

Parameters

hANTemplate	[in] Handle to the ANTemplate object.

Remarks

If *hANTemplate* is NULL, does nothing.

See Also

ANTemplate Module | HANTemplate

6.1.1.6. ANTemplateSaveToFile Function

Saves the ANTemplate to the specified file.

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

Parameters

hANTemplate	[in] Handle to the ANTemplate object.
szFileName	[in] Pointer to a string that specifies file-name.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hANTemplate* or *szFileName* is NULL.
N_E_IO	Input/output error has occurred.

See Also

ANTemplate Module | HANTemplate | ANTemplateCreateFromFile

6.1.1.7. ANTemplateToNFTemplate Function

Converts the ANTemplate to a NFTemplate.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

ANTemplate Module | HANTemplate | HNFTemplate | `ANTemplateCreateFromNFTemplate`

6.1.1.8. ANTemplateToNTemplate Function

Converts the ANTemplate to a NTemplate.

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

Parameters

hANTemplate	[in] Handle to the ANTemplate object.
pHNTemplate	[out] Pointer to a HNTemplate that receives handle to newly created NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

ANTemplate Module | HANTemplate | HNTemplate | `ANTemplateCreateFromNTemplate`

6.2. FMRecord Library

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

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

DLL (Windows): `FMRecord.dll`.

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

Requirements (Windows):

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

Requirements (Linux):

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

Modules

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

6.2.1. FMRecord Module

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

Header file: `FMRecord.h`.

Functions

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

FMRecordAddFingerViewCopy	Adds a copy of the specified finger view to the end of FMRecord finger views.
FMRecordCalculateSize	Calculates the size of a stored FMRecord containing the specified number of finger views of specified size.
FMRecordClearFingerViews	Removes all finger views from the FMRecord.
FMRecordClone	Creates a copy of the FMRecord.
FMRecordCreate	Creates an empty FMRecord.
FMRecordCreateFromMemory	Restores a FMRecord from the specified memory buffer.
FMRecordCreateFromNFRecord	Creates a FMRecord from the specified NFRecord.
FMRecordCreateFromNFTemplate	Creates a FMRecord from the specified NFTemplate.
FMRecordFree	Deletes the FMRecord. After the object is deleted the specified handle is no longer valid.
FMRecordGetCaptureEquipmentCompliance	Retrieves the Capture Equipment Compliance of the FMRecord.
FMRecordGetCaptureEquipmentId	Retrieves the Capture Equipment Id from of the FMRecord.
FMRecordGetCbeffProductId	Retrieves the CBEFF Product Id of the FMRecord.
FMRecordGetFingerView	Retrieves the finger view at the specified index of the FMRecord.
FMRecordGetFingerViewCapacity	Retrieves the number of finger views that the FMRecord can contain.
FMRecordGetFingerViewCount	Retrieves the number of finger views in the FMRecord.
FMRecordGetResolutionX	Retrieves horizontal resolution of the image the FMRecord is made from.
FMRecordGetResolutionY	Retrieves vertical resolution of the image the FMRecord is made from.
FMRecordGetSize	Calculates stored size of the FMRecord.

Copyright © 2006 Neurotechnologija

FMRecordGetSizeX	Retrieves horizontal size of the image the FMRecord is made from.
FMRecordGetSizeY	Retrieves vertical size of the image the FMRecord is made from.
FMRecordInfoDispose	For internal use.
FMRecordRemoveFingerView	Removes the finger view at the specified index of the FMRecord.
FMRecordSaveToMemory	Stores the FMRecord in the specified memory buffer.
FMRecordSetCaptureEquipmentCompliance	Sets the Capture Equipment Compliance of the FMRecord.
FMRecordSetCaptureEquipmentId	Sets the Capture Equipment Id of the FMRecord.
FMRecordSetCbeffProductId	Sets the CBEFF Product Id of the FMRecord.
FMRecordSetFingerViewCapacity	Sets the number of finger views that the FMRecord can contain.
FMRecordToNFTemplate	Converts the FMRecord to a NFTemplate.
FMRecordToNTemplate	Converts the FMRecord to a NTemplate.

Structures

FMRecordInfo	For internal use.

Types

HFMRecord	Handle to FMRecord object.

Macros

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

Copyright © 2006 Neurotechnologija

See Also

FMRecord Library

6.2.1.1. FMRecordAddFingerView Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord | HFmrFingerView | FMRecordGetFingerViewCount

6.2.1.2. FMRecordAddFingerViewCopy Function

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

```
NResult N_API FMRecordAddFingerViewCopy(
```

```
        HFMRecord hRecord,
        HFmrFingerView hSrcFingerView,
        HFmrFingerView * pHFingerView
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord | HFmrFingerView | FMRecordGetFingerViewCount

6.2.1.3. FMRecordCalculateSize Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

FMRecord Module | `FMRecordSaveToMemory` | `FmrFingerViewGetMaxSize`

6.2.1.4. FMRecordClearFingerViews Function

Removes all finger views from the FMRecord.

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

Parameters

hRecord	[in] Handle to the FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

FMRecord Module | HFMRecord

6.2.1.5. FMRecordClone Function

Creates a copy of the FMRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Created object must be deleted using `FMRecordFree` function.

See Also

FMRecord Module | HFMRecord | `FMRecordFree`

6.2.1.6. FMRecordCreate Function

Creates an empty FMRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*sizeX* or *sizeY* is zero. - or -

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

Remarks

Created object must be deleted using FMRecordFree function.

See Also

FMRecord Module | HFMRecord | FMRecordFree

6.2.1.7. FMRecordCreateFromMemory Function

Restores a FMRecord from the specified memory buffer.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

The following flags are supported:

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

Created object must be deleted using `FMRecordFree` function.

See Also

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

6.2.1.8. FMRecordCreateFromNFRecord Function

Creates a FMRecord from the specified NFRecord.

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

Parameters

hNFRecord	[in] Handle to the NFRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.

pHRecord	[out] Pointer to a HFMRecord that receives handle to newly created FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

Created object must be deleted using `FMRecordFree` function.

See Also

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

6.2.1.9. FMRecordCreateFromNFTemplate Function

Creates a FMRecord from the specified NFTemplate.

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

Parameters

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

	handle to newly created FMRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

Created object must be deleted using `FMRecordFree` function.

See Also

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

6.2.1.10. FMRecordFree Function

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

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

Parameters

hRecord	[in] Handle to the FMRecord object.

Remarks

If *hRecord* is NULL, does nothing.

See Also

FMRecord Module | HFMRecord

6.2.1.11. FMRecordGetCaptureEquipmentCompliance Function

Retrieves the Capture Equipment Compliance of the FMRecord.

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

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NByte that receives Capture Equipment Compliance value.

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord | FMRecordSetCaptureEquipmentCompliance

6.2.1.12. FMRecordGetCaptureEquipmentId Function

Retrieves the Capture Equipment Id from of the FMRecord.

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

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NUShort that receives Capture Equipment Id value.

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord | FMRecordSetCaptureEquipmentId

6.2.1.13. FMRecordGetCbeffProductId Function

Retrieves the CBEFF Product Id of the FMRecord.

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

Parameters

hRecord	[in] Handle to the FMRecord object.
pValue	[out] Pointer to NUInt that receives CBEFF Product Id value.

Return Values

If the function succeeds, the return value is N_OK.

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

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

Copyright © 2006 Neurotechnologija

See Also

FMRecord Module | HFMRecord | `FMRecordSetCbeffProductId`

6.2.1.14. FMRecordGetFingerView Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord | `FMRecordGetFingerViewCount`

6.2.1.15. FMRecordGetFingerViewCapacity Function

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

```
NResult N_API FMRecordGetFingerViewCapacity(
```

```
        HFMRecord hRecord,
        NInt * pValue
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

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

6.2.1.16. FMRecordGetFingerViewCount Function

Retrieves the number of finger views in the FMRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

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

6.2.1.17. FMRecordGetResolutionX Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord

6.2.1.18. FMRecordGetResolutionY Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord

Copyright © 2006 Neurotechnologija

6.2.1.19. FMRecordGetSize Function

Calculates stored size of the FMRecord.

```
NResult N_API FMRecordGetSize(
        HFMRecord hRecord,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

FMRecord Module | HFMRecord | `FMRecordSaveToMemory`

6.2.1.20. FMRecordGetSizeX Function

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

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

Copyright © 2006 Neurotechnologija

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord

6.2.1.21. FMRecordGetSizeY Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord

6.2.1.22. FMRecordRemoveFingerView Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FMRecord Module | HFMRecord | `FMRecordGetFingerViewCount`

6.2.1.23. FMRecordSaveToMemory Function

Copyright © 2006 Neurotechnologija

Stores the FMRecord in the specified memory buffer.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the

buffer needed and has the same effect as `FMRecordGetSize` function.

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

The following flags are supported:

- FMR_PROCESS_FIRST_FINGER_VIEW_ONLY
- FMRFV_SKIP_NEUROTEC_FIELDS
- FMRFV_SKIP_RIDGE_COUNTS
- FMRFV_SKIP_SINGULAR_POINTS
- FMRFV_USE_NEUROTEC_FIELDS

See Also

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

6.2.1.24. FMRecordSetCaptureEquipmentCompliance Function

Sets the Capture Equipment Compliance of the FMRecord.

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

Parameters

`hRecord`	[in] Handle to the FMRecord object.
`value`	[in] New Capture Equipment Compliance value.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*`hRecord`* is NULL.

See Also

FMRecord Module | HFMRecord | FMRecordGetCaptureEquipmentCompliance

6.2.1.25. FMRecordSetCaptureEquipmentId Function

Sets the Capture Equipment Id of the FMRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

FMRecord Module | HFMRecord | FMRecordGetCaptureEquipmentId

6.2.1.26. FMRecordSetCbeffProductId Function

Sets the CBEFF Product Id of the FMRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

FMRecord Module | HFMRecord | FMRecordGetCbeffProductId

6.2.1.27. FMRecordSetFingerViewCapacity Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

FMRecord Module | HFMRecord | `FMRecordGetFingerViewCapacity` | `FMRecordGetFingerViewCount`

6.2.1.28. FMRecordToNFTemplate Function

Converts the FMRecord to a NFTemplate.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

See Also

FMRecord Module | HFMRecord | HNFTemplate | `FMRecordCreateFromNFTemplate`

6.2.1.29. FMRecordToNTemplate Function

Converts the FMRecord to a NTemplate.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

See Also

FMRecord Module | HFMRecord | HNTemplate

6.2.2. FmrFingerView Module

Provides functionality for editing and converting ANSI INCITS 378-2004 standard templates finger views (FmrFingerViews) to Neurotechnologija Finger Records (NFRecords).

Header file: `FmrFingerView.h`.

Functions

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

	rFingerView.
FmrFingerViewGetDeltas	Copies all deltas of FmrFingerView to the specified array.
FmrFingerViewGetFingerPosition	Retrieves the finger position of the FmrFingerView.
FmrFingerViewGetFingerQuality	Retrieves the finger quality of the FmrFingerView.
FmrFingerViewGetImpressionType	Retrieves the impression type of the FmrFingerView.
FmrFingerViewGetMaxSize	Retrieves the maximal size of a FmrFingerView in stored FMRecord containing the specified number of minutiae, cores and deltas and the specified ridge counts.
FmrFingerViewGetMaxSizeEx	For internal use.
FmrFingerViewGetMinutia	Retrieves the minutia at the specified index of the FmrFingerView.
FmrFingerViewGetMinutiaCapacity	Retrieves the number of minutiae that the FmrFingerView can contain.
FmrFingerViewGetMinutiaCount	Retrieves the number of minutiae in the FmrFingerView.
FmrFingerViewGetMinutiaEightNeighbour	Retrieves one of the eight minutiae neighbours at the specified index of the minutia at the specified index of the FmrFingerView.
FmrFingerViewGetMinutiaEightNeighbours	Copies all eight minutia neighbours of the minutia at the specified index of the FmrFingerView to the specified array.
FmrFingerViewGetMinutiaFourNeighbour	Retrieves one of the four minutiae neighbours at the specified index of the minutia at the specified index of the FmrFingerView.
FmrFingerViewGetMinutiaFourNeighbours	Copies all four minutia neighbours of the minutia at the specified index of the FmrFingerView to the specified array.
FmrFingerViewGetMinutiae	Copies all minutiae of FmrFingerView to the specified array.
FmrFingerViewGetViewNumber	Retrieves the view number of the FmrFingerView.

FmrFingerViewHasEightNeighbourRidgeCounts	Retrieves a value indicating whether the FmrFingerView has ridge counts to eight neighbours of each minutia.
FmrFingerViewHasFourNeighbourRidgeCounts	Retrieves a value indicating whether the FmrFingerView has ridge counts to four neighbours of each minutia.
FmrFingerViewInsertCore	Inserts a FmrCore into the FmrFingerView at the specified index.
FmrFingerViewInsertDelta	Inserts a FmrDelta into the FmrFingerView at the specified index.
FmrFingerViewInsertMinutia	Inserts a FmrMinutia into the FmrFingerView at the specified index.
FmrFingerViewRemoveCore	Removes the core at the specified index of the FmrFingerView.
FmrFingerViewRemoveDelta	Removes the delta at the specified index of the FmrFingerView.
FmrFingerViewRemoveMinutia	Removes the minutia at the specified index of the FmrFingerView.
FmrFingerViewSetCore	Sets a FmrCore at the specified index of the FmrFingerView.
FmrFingerViewSetCoreCapacity	Sets the number of cores that the FmrFingerView can contain.
FmrFingerViewSetDelta	Sets a FmrDelta at the specified index of the FmrFingerView.
FmrFingerViewSetDeltaCapacity	Sets the number of deltas that the FmrFingerView can contain.
FmrFingerViewSetFingerPosition	Sets the finger position of the FmrFingerView.
FmrFingerViewSetFingerQuality	Sets the finger quality of the FmrFingerView.
FmrFingerViewSetHasEightNeighbourRidgeCounts	Sets a value indicating whether the FmrFingerView has ridge counts to eight neighbours of each minutia.
FmrFingerViewSetHasFourNeighbourRidgeCounts	Sets a value indicating whether the FmrFingerView has ridge counts to four neighbours of each minutia.

Copyright © 2006 Neurotechnologija

FmrFingerViewSetImpression-Type	Sets the impression type of the FmrFinger-View.
FmrFingerViewSetMinutia	Sets a FmrMinutia at the specified index of the FmrFingerView.
FmrFingerViewSetMinutiaCapa-city	Sets the number of minutiae that the Fm-rFingerView can contain.
FmrFingerViewSetMinutiaEight-Neighbour	Sets one of the eight NFMinutiaNeighbours at the specified index of the minutia at the specified index of the FmrFingerView.
FmrFingerViewSetMinu-tiaFourNeighbour	Sets one of the four NFMinutiaNeighbours at the specified index of the minutia at the specified index of the FmrFingerView.
FmrFingerViewSetViewNumber	Sets the view number of the FmrFinger-View.
FmrFingerViewToNFRecord	Converts the FmrFingerView to a NFRe-cord.

Structures

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

Types

HFmrFingerView	Handle to a FmrFingerView object.

Macros

FMRFV_MAX_CORE_COUNT	The maximum number of cores a Fm-rFingerView can contain.
FMRFV_MAX_DELTA_COUNT	The maximum number of deltas a Fm-rFingerView can contain.
FMRFV_MAX_DIMENSION	The maximum value for x and y coordinates of a minutia, core or delta in a FmrFinger-

Copyright © 2006 Neurotechnologija

	View.
FMRFV_MAX_MINUTIA_COUNT	The maximum number of minutiae a FmrFingerView can contain.
FMRFV_NIST_RIDGE_COUNTS	For internal use.
FMRFV_PROCESS_ALL_EXTENDED_DATA	For internal use.
FMRFV_SKIP_NEUROTEC_FIELDS	The flag indicating whether fields of Neurotechnologija should be skipped while loading or saving FmrFingerView.
FMRFV_SKIP_RIDGE_COUNTS	The flag indicating whether ridge counts should be skipped while loading or saving FmrFingerView.
FMRFV_SKIP_SINGULAR_POINTS	The flag indicating whether singular points (cores and deltas) should be skipped while loading or saving FmrFingerView.
FMRFV_USE_NEUROTEC_FIELDS	The flag indicating whether fields of Neurotechnologija should be used while loading or saving FmrFingerView.

See Also

FMRecord Library

6.2.2.1. FmrCore Structure

Represents a core in a FmrFingerView.

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

Fields

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

See Also

FmrFingerView Module

6.2.2.1.1. FmrCore.Angle Field

Angle of this FmrCore.

```
NInt Angle;
```

Remarks

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

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

See Also

FmrCore

6.2.2.1.2. FmrCore.X Field

X coordinate of this FmrCore.

```
NUShort X;
```

Remarks

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

See Also

FmrCore

6.2.2.1.3. FmrCore.Y Field

Y coordinate of this FmrCore.

```
NUShort Y;
```

Remarks

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

See Also

FmrCore

6.2.2.2. FmrDelta Structure

Represents a delta in a FmrFingerView.

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

Fields

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

See Also

FmrFingerView Module

6.2.2.2.1. FmrDelta.Angle1 Field

First angle of this FmrDelta.

```
NInt Angle1;
```

Remarks

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

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

See Also

FmrDelta

6.2.2.2.2. FmrDelta.Angle2 Field

Second angle of this FmrDelta.

```
NInt Angle2;
```

Remarks

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

Copyright © 2006 Neurotechnologija

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

See Also

FmrDelta

6.2.2.2.3. FmrDelta.Angle3 Field

Third angle of this FmrDelta.

```
NInt Angle3;
```

Remarks

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

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

See Also

FmrDelta

6.2.2.2.4. FmrDelta.X Field

X coordinate of this FmrDelta.

```
NUShort X;
```

Remarks

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

See Also

FmrDelta

6.2.2.2.5. FmrDelta.Y Field

Y coordinate of this FmrDelta.

```
NUShort Y;
```

Remarks

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

See Also

FmrDelta

6.2.2.3. FmrFingerViewAddCore Function

Adds a FmrCore to the end of FmrFingerView cores.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrCore

6.2.2.4. FmrFingerViewAddDelta Function

Adds a FmrDelta to the end of FmrFingerView deltas.

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

Copyright © 2006 Neurotechnologija

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrDelta

6.2.2.5. FmrFingerViewAddMinutia Function

Adds a FmrMinutia to the end of FmrFingerView minutiae.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hFingerView* or *pValue* is NULL.
N_E_INVALID_OPERATION	Number of minutiae in FmrFingerView (see FmrFingerViewGetMinutiaCount) is equal to FMR-FV_MAX_MINUTIA_COUNT.

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia

6.2.2.6. FmrFingerViewClearCores Function

Removes all cores from the FmrFingerView.

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

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView

6.2.2.7. FmrFingerViewClearDeltas Function

Removes all deltas from the FmrFingerView.

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

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView

6.2.2.8. FmrFingerViewClearMinutiae Function

Removes all minutiae from the FmrFingerView.

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

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView

6.2.2.9. FmrFingerViewGetCore Function

Retrieves the core at the specified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrCore | `FmrFingerViewGetCoreCount` | `FmrFingerViewSetCore`

Copyright © 2006 Neurotechnologija

6.2.2.10. FmrFingerViewGetCoreCapacity Function

Retrieves the number of cores that the FmrFingerView can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

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

6.2.2.11. FmrFingerViewGetCoreCount Function

Retrieves the number of cores in the FmrFingerView.

```
NResult N_API FmrFingerViewGetCoreCount(
```

```
        HFmrFingerView hFingerView,
        NInt * pValue
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

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

6.2.2.12. FmrFingerViewGetCores Function

Copies all cores of FmrFingerView to the specified array.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

FmrFingerView Module | HFmrFingerView | FmrCore | FmrFingerViewGetCoreCount

6.2.2.13. FmrFingerViewGetDelta Function

Retrieves the delta at the specified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrDelta | FmrFingerViewGetDeltaCount | FmrFingerViewSetDelta

6.2.2.14. FmrFingerViewGetDeltaCapacity Function

Retrieves the number of deltas that the FmrFingerView can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

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

6.2.2.15. FmrFingerViewGetDeltaCount Function

Retrieves the number of deltas in the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

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

See Also

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

6.2.2.16. FmrFingerViewGetDeltas Function

Copies all deltas of FmrFingerView to the specified array.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

FmrFingerView Module | HFmrFingerView | FmrDelta | `FmrFingerViewGetDeltaCount`

6.2.2.17. FmrFingerViewGetFingerPosition Function

Retrieves the finger position of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | NFPosition | FmrFingerViewSetFingerPosition

6.2.2.18. FmrFingerViewGetFingerQuality Function

Retrieves the finger quality of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetFingerQuality

6.2.2.19. FmrFingerViewGetImpressionType Function

Retrieves the impression type of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | NFImpressionType | FmrFingerViewSetImpressionType

Copyright © 2006 Neurotechnologija

6.2.2.20. FmrFingerViewGetMaxSize Function

Retrieves the maximal size of a FmrFingerView in stored FMRecord containing the specified number of minutiae, cores and deltas and the specified ridge counts.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pSize* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaCount* is less than zero or greater than FMRFV_MAX_MINUTIA_COUNT. - or - *coreCount* is less than zero or greater than FMRFV_MAX_CORE_COUNT. - or - *deltaCount* is less than zero or greater than FMRFV_MAX_DELTA_COUNT.

Remarks

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

See Also

FmrFingerView Module

6.2.2.21. FmrFingerViewGetMinutia Function

Retrieves the minutia at the specified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia | FmrFingerViewGetMinutiaCount | FmrFingerViewSetMinutia

6.2.2.22. FmrFingerViewGetMinutiaCapacity Function

Retrieves the number of minutiae that the FmrFingerView can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Minutia capacity is the number of minutiae that the FmrFingerView can store. Minutia count

(see FmrFingerViewGetMinutiaCount function) is the number of minutiae that are actually in the FmrFingerView.

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetMinutiaCapacity | FmrFingerViewGetMinutiaCount

6.2.2.23. FmrFingerViewGetMinutiaCount Function

Retrieves the number of minutiae in the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Minutia capacity (see FmrFingerViewGetMinutiaCapacity and FmrFinger-ViewSetMinutiaCapacity functions) is the number of minutiae that the FmrFinger-View can store. Minutia count is the number of minutiae that are actually in the FmrFinger-View.

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

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

See Also

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

6.2.2.24. FmrFingerViewGetMinutiaEightNeighbour Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Error Code	Condition
	equal to 8.

See Also

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

6.2.2.25. FmrFingerViewGetMinutiaEightNeighbours Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

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

6.2.2.26. FmrFingerViewGetMinutiaFourNeighbour Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

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

6.2.2.27. FmrFingerViewGetMinutiaFourNeighbours Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

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

6.2.2.28. FmrFingerViewGetMinutiae Function

Copies all minutiae of FmrFingerView to the specified array.

```
NResult N_API FmrFingerViewGetMinutiae(
        HFmrFingerView hFingerView,
        FmrMinutia * arMinutiae
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia | `FmrFingerViewGetMinutiaCount`

6.2.2.29. FmrFingerViewGetViewNumber Function

Retrieves the view number of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetViewNumber

6.2.2.30. FmrFingerViewHasEightNeighbourRidgeCounts Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetHasEightNeighbourRidgeCounts

6.2.2.31. FmrFingerViewHasFourNeighbourRidgeCounts Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewSetHasFourNeigh-

bourRidgeCounts

6.2.2.32. FmrFingerViewInsertCore Function

Inserts a FmrCore into the FmrFingerView at the specified index.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrCore | FmrFingerViewGetCore-Count

6.2.2.33. FmrFingerViewInsertDelta Function

Inserts a FmrDelta into the FmrFingerView at the specified index.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

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

6.2.2.34. FmrFingerViewInsertMinutia Function

Inserts a FmrMinutia into the FmrFingerView at the specified index.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia | FmrFingerViewGetMinutiaCount

6.2.2.35. FmrFingerViewRemoveCore Function

Removes the core at the specified index of the FmrFingerView.

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

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

index	[in] Index of core to remove.

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetCoreCount

6.2.2.36. FmrFingerViewRemoveDelta Function

Removes the delta at the specified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetDeltaCount

6.2.2.37. FmrFingerViewRemoveMinutia Function

Removes the minutia at the specified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetMinutiaCount

6.2.2.38. FmrFingerViewSetCore Function

Sets a FmrCore at the specified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrCore | FmrFingerViewGetCoreCount | FmrFingerViewGetCore

6.2.2.39. FmrFingerViewSetCoreCapacity Function

Sets the number of cores that the FmrFingerView can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetCoreCapacity | FmrFingerViewGetCoreCount

6.2.2.40. FmrFingerViewSetDelta Function

Sets a FmrDelta at the specified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

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

6.2.2.41. FmrFingerViewSetDeltaCapacity Function

Sets the number of deltas that the FmrFingerView can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetDeltaCapacity | FmrFingerViewGetDeltaCount

6.2.2.42. FmrFingerViewSetFingerPosition Function

Sets the finger position of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | NFPosition | FmrFingerViewGetFinger-Position

6.2.2.43. FmrFingerViewSetFingerQuality Function

Sets the finger quality of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetFingerQuality

6.2.2.44. FmrFingerViewSetHasEightNeighbourRidgeCounts Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewGetHasEightNeighbourRidgeCounts`

6.2.2.45. FmrFingerViewSetHasFourNeighbourRidgeCounts Function

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

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

Parameters

hFingerView	[in] Handle to the FmrFingerView object.

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | FmrFingerViewGetHasFourNeighbourRidgeCounts

6.2.2.46. FmrFingerViewSetImpressionType Function

Sets the impression type of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | NFImpressionType | FmrFingerViewGetImpressionType

6.2.2.47. FmrFingerViewSetMinutia Function

Sets a FmrMinutia at the specified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | FmrMinutia | FmrFingerViewGetMinutiaCount | FmrFingerViewGetMinutia

6.2.2.48. FmrFingerViewSetMinutiaCapacity Function

Copyright © 2006 Neurotechnologija

Sets the number of minutiae that the FmrFingerView can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

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

6.2.2.49. FmrFingerViewSetMinutiaEightNeighbour Function

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

cified index of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | NFMinutiaNeighbour | FmrFingerViewGetMinutiaCount | FmrFingerViewGetMinutiaEightNeighbour

6.2.2.50. FmrFingerViewSetMinutiaFourNeighbour Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

FmrFingerView Module | HFmrFingerView | NFMinutiaNeighbour | FmrFingerViewGetMinutiaCount | FmrFingerViewGetMinutiaFourNeighbour

Copyright © 2006 Neurotechnologija

6.2.2.51. FmrFingerViewSetViewNumber Function

Sets the view number of the FmrFingerView.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hFingerView* is NULL.

See Also

FmrFingerView Module | HFmrFingerView | `FmrFingerViewGetViewNumber`

6.2.2.52. FmrFingerViewToNFRecord Function

Converts the FmrFingerView to a NFRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

The following flags are supported:

- FMRFV_NIST_RIDGE_COUNTS

See Also

FmrFingerView Module | HFmrFingerView | HNFRecord | FMRecordCreateFromN-FRecord

6.2.2.53. FmrMinutia Structure

Represents a minutia in a FmrFingerView.

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

Fields

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

See Also

FmrFingerView Module

6.2.2.53.1. FmrMinutia.Angle Field

Angle of this FmrMinutia.

```
NByte Angle;
```

Remarks

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

See Also

FmrMinutia

6.2.2.53.2. FmrMinutia.Quality Field

Quality of this FmrMinutia.

```
NByte Quality;
```

Remarks

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

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

See Also

FmrMinutia

6.2.2.53.3. FmrMinutia.Type Field

Type of this FmrMinutia.

```
NFMinutiaType Type;
```

See Also

FmrMinutia

6.2.2.53.4. FmrMinutia.X Field

X coordinate of this FmrMinutia.

```
NUShort X;
```

Copyright © 2006 Neurotechnologija

Remarks

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

See Also

FmrMinutia

6.2.2.53.5. FmrMinutia.Y Field

Y coordinate of this FmrMinutia.

```
NUShort Y;
```

Remarks

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

See Also

FmrMinutia

6.3. NCore Library

Provides infrastructure for Neurotechnologija components.

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

DLL (Windows): `NCore.dll`.

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

Requirements (Windows):

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

Modules

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

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

6.3.1. NCore Module

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

Header file: `NCore.h`.

Functions

`NAlloc`	Allocates memory block.
`NAStrLen`	Gets length of a null-terminated string of ansi chars.
`NCAlloc`	Allocates memory block with all bytes set to zero.
`NCompare`	Compares bytes in two memory blocks.
`NCompareAStr`	Compares two null-terminated strings of ansi chars.
`NCompareStr`	Compares two null-terminated strings of chars.
`NCopy`	Copies data between memory blocks.
`NCopyAStr`	Copies null-terminated string of ansi chars.
`NCopyStr`	Copies null-terminated string of chars.
`NFill`	Sets bytes of memory block to specified value.
`NFree`	Deallocates memory block.
`NMove`	Move data from one memory block to another.
`NReAlloc`	Reallocates memory block.
`NStrLen`	Gets length of a null-terminated string of chars.

Macros

NClear	Clears memory block.

See Also

NCore Library

6.3.2. NErrors Module

Defines error codes used in Neurotechnologija components.

Header file: `NErrors.h`.

Macros

-10	N_E_ARGUMENT	Argument is invalid.
-11	N_E_ARGUMENT_NULL	Argument value is NULL where non-NULL value was expected.
-12	N_E_ARGUMENT_OUT_OF_RANGE	Argument value is out of range.
-2	N_E_CORE	Standard error has occurred (for internal use).
-15	N_E_END_OF_STREAM	Attempted to read file or buffer after its end.
-1	N_E_FAILED	Unspecified error has occurred.
-13	N_E_FORMAT	Format of argument value is invalid.
-9	N_E_INDEX_OUT_OF_RANGE	Index is out of range (for internal use).
-7	N_E_INVALID_OPERATION	Attempted to perform invalid operation.
-14	N_E_IO	Input/output error has occurred.
-201	N_E_LM_CONNECTION	Error during connection with LAN License Manager has occurred.
-202	N_E_LM_NO_MORE_LICENSE	Library has not been registered because registered license count achieved LAN License Manager license limit.
-5	N_E_NOT_IMPLEMENTED	Functionality is not implemented.

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

See Also

NCore Library

6.3.3. NParameters Module

Provides functionality for work with parameters for Neurotechnologija components.

Header file: `NParameters.h`.

Macros

N_PC_TYPE_ID	Specifies that type id (NInt value, one of N_TYPE_XXX) of the parameter should be retrieved.
NParameterMakeId	Makes parameter id.
N_TYPE_BOOL	Specifies that parameter type is NBool.
N_TYPE_BYTE	Specifies that parameter type is NByte.
N_TYPE_CHAR	Specifies that parameter type is NChar.
N_TYPE_DOUBLE	Specifies that parameter type is NDouble.
N_TYPE_FLOAT	Specifies that parameter type is NFloat.

Copyright © 2006 Neurotechnologija

N_TYPE_INT	Specifies that parameter type is NInt.
N_TYPE_LONG	Specifies that parameter type is NLong.
N_TYPE_SBYTE	Specifies that parameter type is NSByte.
N_TYPE_SHORT	Specifies that parameter type is NShort.
N_TYPE_STRING	Specifies that parameter type is null-terminated string of NChar.
N_TYPE_UINT	Specifies that parameter type is NUInt.
N_TYPE_ULONG	Specifies that parameter type is NULong.
N_TYPE_USHORT	Specifies that parameter type is NUShort.

See Also

NCore Library

6.3.3.1. NParameterMakeId Macro

Makes parameter id.

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

Parameters

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

See Also

NParameters Module

6.3.4. NTypes Module

Defines types and macros used in Neurotechnologija components.

Header file: `NTypes.h`.

Structures

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

Enumerations

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

Types

NAChar	ANSI character (8-bit).
NBool	Same as NBoolean.
NBoolean	32-bit boolean value. See also NTrue and NFalse.
NByte	Same as NUInt8.
NChar	Character type (same as NAChar).
NDouble	Double precision floating point number.
NFloat	Same as NSingle.
NHandle	Pointer to unspecified data (same as void *).
NInt	Same as NInt32.
NInt8	8-bit signed integer (signed byte).
NInt16	16-bit signed integer (short).
NInt32	32-bit signed integer (int).
NInt64	64-bit signed integer (long). Not available on some 32-bit platforms.
NLong	Same as NInt64.
NPosType	Platform dependent position type. Signed 64-bit (or 32-bit on some platforms) integer on 32-bit platform, signed 64-bit integer on

Copyright © 2006 Neurotechnologija

	64-bit platform).
NResult	Result of a function (same as NInt). See also NErrors module.
NSByte	Same as NInt8.
NShort	Same as NInt16.
NSingle	Single precision floating point number.
NSizeType	Platform dependent size type. Unsigned 32-bit integer on 32-bit platform, unsigned 64-bit integer on 64-bit platform.
NUInt	Same as NUInt32.
NUInt8	8-bit unsigned integer (byte).
NUInt16	16-bit unsigned integer (unsigned short).
NUInt32	32-bit unsigned integer (unsigned int).
NUInt64	64-bit unsigned integer (unsigned long). Not available on some 32-bit platforms.
NULong	Same as NUInt64.
NUShort	Same as NUInt16.

Macros

N_64	Defined if compiling for 64-bit architecture.
N_ANSI_C	Defined if ANSI C language compliance is enabled in compiler.
N_API	Defines functions calling convention (stdcall on Windows).
N_BIG_ENDIAN	Defined if compiling for big-endian processor architecture.
N_BYTE_MAX	Maximum value for NByte.
N_BYTE_MIN	Minimum value for NByte.
N_CALLBACK	Defined callbacks calling convention (stdcall on Windows).
N_CPP	Defined if compiling as C++ code.

Copyright © 2006 Neurotechnologija

N_DECLARE_HANDLE	Declares handle with specified name.
N_DOUBLE_MAX	Maximum value for NDouble.
N_DOUBLE_MIN	Minimum value for NDouble.
N_GCC	Defined if compiling with GCC.
N_FLOAT_MAX	Maximum value for NFloat.
N_FLOAT_MIN	Minimum value for NFloat.
N_INT_MAX	Maximum value for NInt.
N_INT_MIN	Minimum value for NInt.
N_INT8_MAX	Maximum value for NInt8.
N_INT8_MIN	Minimum value for NInt8.
N_INT16_MAX	Maximum value for NInt16.
N_INT16_MIN	Minimum value for NInt16.
N_INT32_MAX	Maximum value for NInt32.
N_INT32_MIN	Minimum value for NInt32.
N_INT64_MAX	Maximum value for NInt64.
N_INT64_MIN	Minimum value for NInt64.
N_LONG_MAX	Maximum value for NLong.
N_LONG_MIN	Minimum value for NLong.
N_MSVC	Defined if compiling with Microsoft Visual C++.
N_NO_FLOAT	Defined if compiling for platform without floating-point support.
N_NO_INT_64	Defined if compiling for platform without 64-bit integer types support.
N_POS_TYPE_MIN	Minimum value for NPosType.
N_POS_TYPE_MAX	Maximum value for NPosType.
N_SBYTE_MAX	Maximum value for NSByte.
N_SBYTE_MIN	Minimum value for NSByte.
N_SHORT_MAX	Maximum value for NShort.

Copyright © 2006 Neurotechnologija

N_SHORT_MIN	Minimum value for NShort.
N_SINGLE_MAX	Maximum value for NSingle.
N_SINGLE_MIN	Minimum value for NSingle.
N_SIZE_TYPE_MIN	Minimum value for NSizeType.
N_SIZE_TYPE_MAX	Maximum value for NSizeType.
N_UINT_MAX	Maximum value for NUInt.
N_UINT_MIN	Minimum value for NUInt.
N_UINT8_MAX	Maximum value for NUInt8.
N_UINT8_MIN	Minimum value for NUInt8.
N_UINT16_MAX	Maximum value for NUInt16.
N_UINT16_MIN	Minimum value for NUInt16.
N_UINT32_MAX	Maximum value for NUInt32.
N_UINT32_MIN	Minimum value for NUInt32.
N_UINT64_MAX	Maximum value for NUInt64.
N_UINT64_MIN	Minimum value for NUInt64.
N_ULONG_MAX	Maximum value for NULong.
N_ULONG_MIN	Minimum value for NULong.
N_USHORT_MAX	Maximum value for NUShort.
N_USHORT_MIN	Minimum value for NUShort.
N_WINDOWS	Defined if compiling for Windows.
NULL	Null value for pointer.
NFalse	False value for NBoolean.
NIsReverseByteOrder	Checks if specified byte order is reverse to system byte order.
NTrue	True value for NBoolean.

See Also

NCore Library

Copyright © 2006 Neurotechnologija

6.3.4.1. NByteOrder Enumeration

Specifies byte order.

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

Members

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

See Also

NTypes Module

6.3.4.2. NFileAccess Enumeration

Specifies access to a file.

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

Members

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

See Also

NTypes Module

6.3.4.3. NIndexPair Structure

Represents a pair of indexes.

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

Fields

Copyright © 2006 Neurotechnologija

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

See Also

NTypes Module

6.3.4.3.1. NIndexPair.Index1 Field

First index of this NIndexPair.

```
NInt Index1;
```

See Also

NIndexPair

6.3.4.3.2. NIndexPair.Index2 Field

Second index of this NIndexPair.

```
NInt Index2;
```

See Also

NIndexPair

6.3.4.4. NRational Structure

Represents a signed rational number.

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

Fields

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

See Also

NTypes Module

6.3.4.4.1. NRational.Denominator Field

Denominator of this NRational.

```
NInt Denominator;
```

See Also

NRational

6.3.4.4.2. NRational.Numerator Field

Numerator of this NRational.

```
NInt Numerator;
```

See Also

NRational

6.3.4.5. NURational Structure

Represents an unsigned rational number.

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

Fields

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

See Also

NTypes Module

6.3.4.5.1. NURational.Denominator Field

Denominator of this NURational.

```
NUInt Denominator;
```

See Also

NURational

6.3.4.5.2. NURational.Numerator Field

Numerator of this NURational.

```
NUInt Numerator;
```

See Also

NURational

6.4. NFExtractor Library

Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using MegaMatcher algorithm.

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

DLL (Windows): `NFExtractor.dll`.

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

Requirements (Windows):

- `NCore.dll`.
- `NFRecord.dll`.

Requirements (Linux):

- `libNCore.so`.
- `libNFRecord.so`.

Modules

NFExtractor	Provides functionality for extracting Neurotechnologija Finger Records (NFRecords) from fingerprint images using MegaMatcher algorithm encapsulated in Neurotechnologija Fingerprint Features Extractor (NFExtractor) object.

6.4.1. NFExtractor Module

Provides functionality for extracting Neurotechnologija Finger Records (NFRecords) from fingerprint images using MegaMatcher algorithm encapsulated in Neurotechnologija Fingerprint Features Extractor (NFExtractor) object.

Header file: `NFExtractor.h` (includes `NFExtractorParams.h` and `NFExtractorTypes.h`)

Functions

`NfeCopyParameters`	Copies parameter values from one NFExtractor to another.
`NfeCreate`	Creates a NFExtractor.
`NfeExtract`	Extracts a packed NFRecord from the image using the specified NFExtractor.
`NfeExtractFromImage`	Extracts a packed NFRecord from the NImage using the specified NFExtractor.
`NfeExtractUnpacked`	Extracts a NFRecord from the image using the specified NFExtractor.
`NfeExtractUnpackedFromImage`	Extracts a NFRecord from the NImage using the specified NFExtractor.
`NfeFree`	Deletes the NFExtractor. After the object is deleted the specified handle is no longer valid.
`NfeGetMaxTemplateSize`	Retrieves maximal size of packed NFRecord the specified NFExtractor can extract.
`NfeGetParameter`	Retrieves value of the specified parameter of the specified NFExtractor.
`NfeIsRegistered`	Checks if NFExtractor library is registered.
`NfeRegister`	Registers NFExtractor library.
`NfeReset`	Sets default values for all parameters of the specified NFExtractor.
`NfeSetParameter`	Sets value of the specified parameter of the specified NFExtractor.
`NfeUnregister`	Unregisters NFExtractor library.
`NfeUpdateTemplate`	Updates the packed NFRecord using the specified image and the specified NFExtractor.
`NfeUpdateTemplateFromImage`	Updates the packed NFRecord using the specified NImage and the specified NFExtractor.

Copyright © 2006 Neurotechnologija

NfeUpdateUnpackedTemplate	Updates the NFRecord using the specified image and the specified NFExtractor.
NfeUpdateUnpackedTemplateFromImage	Updates the NFRecord using the specified NImage and the specified NFExtractor.

Enumerations

NfeReturnedImage	Specifies kind of image returned after extraction using NFExtractor.

Types

HNFExtractor	Handle to NFExtractor object.

Macros

NFE_MODE_ATMEL_FINGERCHIP	The mode for Atmel FingerChip sensor.
NFE_MODE_BIOMETRIKA_FX2000	The mode for Biometrika FX2000 scanner.
NFE_MODE_BMF_BLP100	The mode for BMF BLP100 scanner.
NFE_MODE_CROSSMATCH_VERIFIER 300	The mode for CrossMatch Verifier 300 scanner.
NFE_MODE_DIGITALPERSONA_UARE U	The mode for Digital Persona U.are.U scanner.
NFE_MODE_GENERAL	The general mode.
NFE_MODE_IDENTICATORTECHNOLOGY_DF90	The mode for Identicator Technology DF90 scanner.
NFE_MODE_IDENTIX_DFR2090	The mode for Identix DFR-2090 scanner.
NFE_MODE_IDENTIX_TOUCHVIEW	The mode for Identix TouchView scanner.
NFE_MODE_KEYTRONIC_SECUREDESKTOP	The mode for Keytronic Secure Desktop scanner.
NFE_MODE_PRECISEBIOMETRICS_100CS	The mode for Precise Biometrics 100CS scanner.
NFE_MODE_SECUGEN_HAMSTER	The mode for Secugen Hamster scanner.

Copyright © 2006 Neurotechnologija

NFE_MODE_STMICROELECTRONICS_TOUCHCHIP	The mode for STMicroelectronics Touch-Chip sensor.
NFEP_COPYRIGHT	Identifier specifying library copyright static read-only parameter of type N_TYPE_STRING.
NFEP_EXTRACTED_RIDGE_COUNTS	Identifier specifying type of ridge counts extracted parameter of type N_TYPE_INT. Parameter value can be one of the NFRidgeCountsType enumeration members.
NFEP_MODE	Identifier specifying mode (parameter value set) parameter of type N_TYPE_UINT. Parameter value can be one of the NFE_MODE_XXX.
NFEP_NAME	Identifier specifying library name static read-only parameter of type N_TYPE_STRING.
NFEP_RETURNED_IMAGE	Identifier specifying kind of image returned after extraction parameter of type N_TYPE_INT. Parameter value can be one of the NfeReturnedImage enumeration members.
NFEP_VERSION_HIGH	Identifier specifying high part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
NFEP_VERSION_LOW	Identifier specifying low part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

See Also

NFExtractor Library

6.4.1.1. NfeCopyParameters Function

Copies parameter values from one NFExtractor to another.

```
NResult N_API NfeCopyParameters(
```

```
        HNFExtractor hDstExtractor,
        HNFExtractor hSrcExtractor
);
```

Parameters

hDstExtractor	[in] Handle to the destination NFExtractor object.
hSrcExtractor	[in] Handle to the source NFExtractor object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hDstExtractor* or *hSrcExtractor* is NULL.

See Also

NFExtractor Module | HNFExtractor

6.4.1.2. NfeCreate Function

Creates a NFExtractor.

```
NResult N_API NfeCreate(
        HNFExtractor * pHExtractor
);
```

Parameters

pHExtractor	[out] Pointer to HNFExtractor that receives handle to created NFExtractor object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHExtractor* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NfeFree` function.

See Also

NFExtractor Module | HNFExtractor | `NfeFree`

6.4.1.3. NfeExtract Function

Extracts a packed NFRecord from the image using the specified NFExtractor.

```
NResult N_API NfeExtract(
        HNFExtractor hExtractor,
        NUShort width,
        NUShort height,
        NSizeType stride,
        NUShort horzResolution,
        NUShort vertResolution,
        NByte * pixels,
        NFPosition position,
        NFImpressionType impressionType,
        void * buffer,
        NSizeType bufferSize,
        NSizeType * pSize,
        NBool * pIsLowQuality
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.
width	[in] Specifies image width.
height	[in] Specifies image height.
stride	[in] Specifies length of the image row in bytes.
horzResolution	[in] Specifies horizontal resolution of the image.

vertResolution	[in] Specifies vertical resolution of the image.
pixels	[in, out] Pointer to memory block containing image pixels.
position	[in] Specifies finger position.
impressionType	[in] Specifies impression type.
buffer	[out] Pointer to memory buffer to store extracted packed NFRecord.
bufferSize	[in] Size of memory buffer to store extracted packed NFRecord.
pSize	[out] Pointer to NSizeType that receives size of extracted packed NFRecord.
pIsLowQuality	[out] NTrue if fingerprint image is of low quality, NFalse otherwise.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*stride* is less than *width*. - or - *position* is invalid. - or - *impressionType* is invalid. - or - *bufferSize* is less than needed to store extracted packed NFRecord.
N_E_ARGUMENT_NULL	*hExtractor*, *pixels*, *buffer*, *pSize*, or *pIsLowQuality* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero.

Copyright © 2006 Neurotechnologija

Error Code	Condition
	- or - *horzResolution* or *vertResolution* is out of supported range.

Remarks

position and *impressionType* are written to extracted NFRecord.

Memory for *buffer* must be allocated before calling the function. NfeGetMaxTemplateSize function should be used to learn the size needed for allocation.

See Also

NFExtractor Module | HNFExtractor | NfeGetMaxTemplateSize | NfeExtractUnpacked | NfeExtractFromImage

6.4.1.4. NfeExtractFromImage Function

Extracts a packed NFRecord from the NImage using the specified NFExtractor.

```
NResult N_API NfeExtractFromImage(
        HNFExtractor hExtractor,
        HNImage hImage,
        NFPosition position,
        NFImpressionType impressionType,
        void * buffer,
        NSizeType bufferSize,
        NSizeType * pSize,
        NBool * pIsLowQuality
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.
hImage	[in] Handle to the image.
position	[in] Specifies finger position.
impressionType	[in] Specifies impression type.
buffer	[out] Pointer to memory buffer to store extracted packed NFRecord.
bufferSize	[in] Size of memory buffer to store extracted packed NFRecord.

Copyright © 2006 Neurotechnologija

pSize	[out] Pointer to NSizeType that receives size of extracted packed NFRecord.
pIsLowQuality	[out] NTrue if fingerprint image is of low quality, NFalse otherwise.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*hImage* is invalid. - or - *position* is invalid. - or - *impressionType* is invalid. - or - *bufferSize* is less than needed to store extracted packed NFRecord.
N_E_ARGUMENT_NULL	*hExtractor*, *hImage*, *buffer*, *pSize*, or *pIsLowQuality* is NULL.

Remarks

position and *impressionType* are written to extracted NFRecord.

Memory for *buffer* must be allocated before calling the function. NfeGetMaxTemplateSize function should be used to learn the size needed for allocation.

See Also

NFExtractor Module | HNFExtractor | HNImage | NfeGetMaxTemplateSize | NfeExtract | NfeExtractUnpackedFromImage

6.4.1.5. NfeExtractUnpacked Function

Extracts a NFRecord from the image using the specified NFExtractor.

```
NResult N_API NfeExtractUnpacked(
        HNFExtractor hExtractor,
        NUShort width,
        NUShort height,
        NSizeType stride,
        NUShort horzResolution,
        NUShort vertResolution,
        NByte * pixels,
        NFPosition position,
        NFImpressionType impressionType,
        HNFRecord * pHRecord,
        NBool * pIsLowQuality
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.
width	[in] Specifies image width.
height	[in] Specifies image height.
stride	[in] Specifies length of the image row in bytes.
horzResolution	[in] Specifies horizontal resolution of the image.
vertResolution	[in] Specifies vertical resolution of the image.
pixels	[in, out] Pointer to memory block containing image pixels.
position	[in] Specifies finger position.
impressionType	[in] Specifies impression type.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object containing extracted template.
pIsLowQuality	[out] NTrue if fingerprint image is of low quality, NFalse otherwise.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*stride* is less than *width*. - or - *position* is invalid. - or - *impressionType* is invalid.
N_E_ARGUMENT_NULL	*hExtractor*, *pixels*, *pHRecord*, or *pIsLowQuality* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is out of supported range.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

NFExtractor Module | HNFExtractor | HNFRecord | NfeExtract | NfeExtractUnpackedFromImage

6.4.1.6. NfeExtractUnpackedFromImage Function

Extracts a NFRecord from the NImage using the specified NFExtractor.

```
NResult N_API NfeExtractUnpackedFromImage(
        HNFExtractor hExtractor,
        HNImage hImage,
        NFPosition position,
        NFImpressionType impressionType,
        HNImage * pHRecord,
        NBool * pIsLowQuality
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object.
hImage	[in] Handle to the image.

position	[in] Specifies finger position.
impressionType	[in] Specifies impression type.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object containing extracted template.
pIsLowQuality	[out] NTrue if fingerprint image is of low quality, NFalse otherwise.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*hImage* is invalid. - or - *position* is invalid. - or - *impressionType* is invalid.
N_E_ARGUMENT_NULL	*hExtractor*, *hImage*, *pHRecord*, or *pIsLowQuality* is NULL.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

NFExtractor Module | HNFExtractor | HNImage | HNFRecord | NfeExtractFromImage | NfeExtractUnpacked

6.4.1.7. NfeFree Function

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

```
void N_API NfeFree(
        HNFExtractor hExtractor
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.

Remarks

If *hExtractor* is NULL, does nothing.

See Also

NFExtractor Module | HNFExtractor

6.4.1.8. NfeGetMaxTemplateSize Function

Retrieves maximal size of packed NFRecord the specified NFExtractor can extract.

```
NResult N_API NfeGetMaxTemplateSize(
        HNFExtractor hExtractor,
        NSizeType * pSize
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.
pSize	[out] Pointer to NSizeType that receives maximal template size.

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFExtractor Module | HNFExtractor

6.4.1.9. NfeGetParameter Function

Retrieves value of the specified parameter of the specified NFExtractor.

```
NResult N_API NfeGetParameter(
        HNFExtractor hExtractor,
        NUInt parameterId,
        void * pValue
);
```

Parameters

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

Return Values

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

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

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

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

Remarks

The following values can be used for *parameterId*:

- NFEP_COPYRIGHT

- NFEP_EXTRACTED_RIDGE_COUNTS
- NFEP_MODE
- NFEP_NAME
- NFEP_RETURNED_IMAGE
- NFEP_VERSION_HIGH
- NFEP_VERSION_LOW

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

See Also

NFExtractor Module | HNFExtractor | NfeSetParameter

6.4.1.10. NfeIsRegistered Function

Checks if NFExtractor library is registered.

```
NBool N_API NfeIsRegistered(void);
```

Return Values

NTrue if library is registered, NFalse otherwise.

See Also

NFExtractor Module | NfeRegister | NfeUnregister

6.4.1.11. NfeRegister Function

Registers NFExtractor library.

```
NResult N_API NfeRegister(
        const NChar * szServer
);
```

Parameters

szServer	[in] Pointer to string that specifies license manager server name or IP address.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*szServer* is NULL.
N_E_LM_CONNECTION	There was error during connection or communication with license manager server.
N_E_LM_NO_MORE_LICENSE	Registered license count achieved license manager license limit for NFExtractor library.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using `NfeUnregister` function.

See Also

NFExtractor Module | `NfeUnregister` | `NfeIsRegistered`

6.4.1.12. NfeReset Function

Sets default values for all parameters of the specified NFExtractor.

```
NResult N_API NfeReset(
        HNFExtractor hExtractor
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hExtractor* is NULL.

See Also

NFExtractor Module | HNFExtractor

6.4.1.13. NfeReturnedImage Enumeration

Specifies kind of image returned after extraction using NFExtractor.

```
typedef enum NfeReturnedImage_ { } NfeReturnedImage;
```

Members

`nferiBinarized`	Binarized (filtered) image is written to the image used for extraction.
`nferiNone`	The image used for extraction is left un-changed.
`nferiSkeletonized`	Skeletonized image is written to the image used for extraction.

See Also

NFExtractor Module

6.4.1.14. NfeSetParameter Function

Sets value of the specified parameter of the specified NFExtractor.

```
NResult N_API NfeSetParameter(
        HNFExtractor hExtractor,
        NUInt parameterId,
        const void * pValue
);
```

Parameters

`hExtractor`	[in] Handle to the NFExtractor object. Can be NULL if setting static parameter value.
`parameterId`	[in] Identifier of the parameter to set.
`pValue`	[in] Pointer to the parameter value to set.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*parameterId* specifies a non-static parameter and *hExtractor* is NULL. - or - *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	Value *pValue* points to is out of range.
N_E_PARAMETER	*parameterId* is invalid.
N_E_PARAMETER_READ_ONLY	*parameterId* specifies read-only parameter.

Remarks

The following values can be used for *parameterId*:

- NFEP_EXTRACTED_RIDGE_COUNTS
- NFEP_MODE
- NFEP_RETURNED_IMAGE

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

See Also

NFExtractor Module | HNFExtractor | `NfeGetParameter`

6.4.1.15. NfeUnregister Function

Unregisters NFExtractor library.

```
void N_API NfeUnregister(void);
```

See Also

NFExtractor Module | NfeRegister | NfeIsRegistered

6.4.1.16. NfeUpdateTemplate Function

Updates the packed NFRecord using the specified image and the specified NFExtractor.

```
NResult N_API NfeUpdateTemplate(
        HNFExtractor hExtractor,
        NUShort width,
        NUShort height,
        NSizeType stride,
        NUShort horzResolution,
        NUShort vertResolution,
        NByte * pixels,
        void * templ,
        NSizeType templSize
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.
width	[in] Specifies image width.
height	[in] Specifies image height.
stride	[in] Specifies length of the image row in bytes.
horzResolution	[in] Specifies horizontal resolution of the image.
vertResolution	[in] Specifies vertical resolution of the image.
pixels	[in, out] Pointer to memory block containing image pixels.
templ	[in, out] Pointer to memory buffer containing packed NFRecord.
templSize	[in] Size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*stride* is less than *width*.
N_E_ARGUMENT_NULL	*hExtractor*, *pixels*, or *templ* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is out of supported range.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NFRecord format.

Remarks

This function updates ridge counts and minutiae curvatures and Gs using the image provided.

See Also

NFExtractor Module | HNFExtractor | NfeUpdateUnpackedTemplate | NfeUpdateTemplateFromImage

6.4.1.17. NfeUpdateTemplateFromImage Function

Updates the packed NFRecord using the specified NImage and the specified NFExtractor.

```
NResult N_API NfeUpdateTemplateFromImage(
        HNFExtractor hExtractor,
        HNImage hImage,
        void * templ,
        NSizeType templSize
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.
hImage	[in] Handle to the image.
templ	[in, out] Pointer to memory buffer containing packed NFRecord.
templSize	[in] Size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*hImage* is invalid.
N_E_ARGUMENT_NULL	*hExtractor*, *hImage*, or *templ* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NFRecord format.

Remarks

This function updates ridge counts and minutiae curvatures and Gs using the image provided.

See Also

NFExtractor Module | HNFExtractor | HNImage | NfeUpdateTemplate | NfeUpdateUnpackedTemplateFromImage

6.4.1.18. NfeUpdateUnpackedTemplate Function

Updates the NFRecord using the specified image and the specified NFExtractor.

```
NResult N_API NfeUpdateUnpackedTemplate(
        HNFExtractor hExtractor,
        NUShort width,
        NUShort height,
        NSizeType stride,
        NUShort horzResolution,
        NUShort vertResolution,
        NByte * pixels,
        HNImage hRecord
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.
width	[in] Specifies image width.
height	[in] Specifies image height.

stride	[in] Specifies length of the image row in bytes.
horzResolution	[in] Specifies horizontal resolution of the image.
vertResolution	[in] Specifies vertical resolution of the image.
pixels	[in, out] Pointer to memory block containing image pixels.
hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*stride* is less than *width*.
N_E_ARGUMENT_NULL	*hExtractor*, *pixels*, or *pHRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is out of supported range.

Remarks

This function updates ridge counts and minutiae curvatures and Gs using the image provided.

See Also

NFExtractor Module | HNFExtractor | HNFRecord | NfeUpdateTemplate | NfeUpdateUnpackedTemplateFromImage

6.4.1.19. NfeUpdateUnpackedTemplateFromImage Function

Updates the NFRecord using the specified NImage and the specified NFExtractor.

```
NResult N_API NfeUpdateUnpackedTemplateFromImage(
```

```
        HNFExtractor hExtractor,
        HNImage hImage,
        HNImage hRecord
);
```

Parameters

hExtractor	[in] Handle to the NFExtractor object.
hImage	[in] Handle to the image.
hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT	*hImage* is invalid.
N_E_ARGUMENT_NULL	*hExtractor*, *hImage*, or *pHRecord* is NULL.

Remarks

This function updates ridge counts and minutiae curvatures and Gs using the image provided.

See Also

NFExtractor Module | HNFExtractor | HNImage | HNFRecord | NfeUpdateTemplate-FromImage | NfeUpdateUnpackedTemplate

6.5. NFMatcher Library

Provides functionality for comparing Neurotechnologija Finger Records using MegaMatcher algorithm.

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

DLL (Windows): NFMatcher.dll.

Shared object (Linux): libNFMatcher.so.

Requirements (Windows):

- `NCore.dll`.
- `NFRecord.dll`.

Requirements (Linux):

- `libNCore.so`.
- `libNFRecord.so`.

Modules

NFMatcher	Provides functionality for comparing Neurotechnologija Finger Records (NFRecords) using MegaMatcher algorithm encapsulated in Neurotechnologija Finger Matcher (NFMatcher) object.

6.5.1. NFMatcher Module

Provides functionality for comparing Neurotechnologija Finger Records (NFRecords) using MegaMatcher algorithm encapsulated in Neurotechnologija Finger Matcher (NFMatcher) object.

Header file: `NFMatcher.h` (includes `NFMatcherParams.h` and `NfmMatchDetails.h`).

Functions

`NfmCopyParameters`	Copies parameter values from one NFMatcher to another.
`NfmCreate`	Creates a NFMatcher.
`NfmFree`	Deletes the NFMatcher. After the object is deleted the specified handle is no longer valid.
`NfmGetParameter`	Retrieves value of the specified parameter of the specified NFMatcher.
`NfmIdentifyEnd`	Ends identification using the specified NFMatcher.
`NfmIdentifyNext`	Compares the specified packed NFRecord with the one identification was started with using the specified NFMatcher.
`NfmIdentifyStart`	Starts identification with the specified

Copyright © 2006 Neurotechnologija

	packed NFRecord using the specified NFMatcher.
NfmIsRegistered	Checks if NFMatcher library is registered.
NfmMatchDetailsFree	Deletes NfmMatchDetails.
NfmRegister	Registers NFMatcher library.
NfmReset	Sets default values for all parameters of the specified NFMatcher.
NfmSetParameter	Sets value of the specified parameter of the specified NFMatcher.
NfmUnregister	Unregisters NFMatcher library.
NfmVerify	Compares two packed NFRecords using the specified NFMatcher.

Structures

NfmMatchDetails	Represents details of matching performed with NFMatcher.

Enumerations

NfmSpeed	Specifies matching speed used in NFMatcher.

Types

HNFMatcher	Handle to NFMatcher object.

Macros

NFM_MODE_ATMEL_FINGERCHIP	The mode for Atmel FingerChip sensor.
NFM_MODE_BIOMETRIKA_FX2000	The mode for Biometrika FX2000 scanner.
NFM_MODE_BMF_BLP100	The mode for BMF BLP100 scanner.
NFM_MODE_CROSSMATCH_VERIFIER 300	The mode for CrossMatch Verifier 300 scanner.

Copyright © 2006 Neurotechnologija

NFM_MODE_DIGITALPERSONA_UARE U	The mode for Digital Persona U.are.U scanner.
NFM_MODE_GENERAL	The general mode.
NFM_MODE_IDENTICATORTECHNOL OGY_DF90	The mode for Identicator Technology DF90 scanner.
NFM_MODE_IDENTIX_DFR2090	The mode for Identix DFR-2090 scanner.
NFM_MODE_IDENTIX_TOUCHVIEW	The mode for Identix TouchView scanner.
NFM_MODE_KEYTRONIC_SECUREDE SKTOP	The mode for Keytronic Secure Desktop scanner.
NFM_MODE_PRECISEBIOMETRICS_10 0CS	The mode for Precise Biometrics 100CS scanner.
NFM_MODE_SECUGEN_HAMSTER	The mode for Secugen Hamster scanner.
NFM_MODE_STMICROELECTRONICS_ TOUCHCHIP	The mode for STMicroelectronics TouchChip sensor.
NFMP_COPYRIGHT	Identifier specifying library copyright static read-only parameter of type N_TYPE_STRING.
NFMP_MATCHING_SPEED	Identifier specifying matching speed parameter of type N_TYPE_INT. Parameter value can be one of the NfmSpeed enumeration members.
NFMP_MATCHING_THRESHOLD	Identifier specifying matching threshold (biggest allowed FAR) parameter of type N_TYPE_INT. Parameter value is equal to -12 * log10(FAR) and must be not less than zero (for example, 48 for FAR = 0.01%).
NFMP_MAXIMAL_ROTATION	Identifier specifying modulus of maximal rotation allowed between two matched NFRecords parameter of type N_TYPE_BYTE. Parameter value is specified in 180/128 degrees units and can not be greater than 128 (180 degrees).
NFMP_MODE	Identifier specifying mode (parameter value set) parameter of type N_TYPE_UINT. Parameter value can be one of the NFM_MODE_XXX.
NFMP_NAME	Identifier specifying library name static read-only parameter of type

	N_TYPE_STRING.
NFMP_VERSION_HIGH	Identifier specifying high part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
NFMP_VERSION_LOW	Identifier specifying low part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

See Also

NFMatcher Library

6.5.1.1. NfmCopyParameters Function

Copies parameter values from one NFMatcher to another.

```
NResult N_API NfmCopyParameters(
        HNFMatcher hDstMatcher,
        HNFMatcher hSrcMatcher
);
```

Parameters

hDstMatcher	[in] Handle to the destination NFMatcher object.
hSrcMatcher	[in] Handle to the source NFMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hDstMatcher* or *hSrcMatcher* is NULL.

Error Code	Condition
N_E_INVALID_OPERATION	Identification is started on *hDstMatcher*.

See Also

NFMatcher Module | HNFMatcher

6.5.1.2. NfmCreate Function

Creates a NFMatcher.

```
NResult N_API NfmCreate(
        HNFMatcher * pHMatcher
);
```

Parameters

pHMatcher	[out] Pointer to HNFMatcher that receives handle to created NFMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*pHMatcher* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NfmFree` function.

See Also

NFMatcher Module | HNFMatcher | `NfmFree`

6.5.1.3. NfmFree Function

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

Copyright © 2006 Neurotechnologija

```
void N_API NfmFree(
        HNFMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to NFMatcher object.

Remarks

If *hMatcher* is NULL, does nothing.

See Also

NFMatcher Module | HNFMatcher

6.5.1.4. NfmGetParameter Function

Retrieves value of the specified parameter of the specified NFMatcher.

```
NResult N_API NfmGetParameter(
        HNFMatcher hMatcher,
        NUInt parameterId,
        void * pValue
);
```

Parameters

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

Return Values

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

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

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

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

Remarks

The following values can be used for *parameterId*:

- NFMP_COPYRIGHT
- NFMP_MATCHING_SPEED
- NFMP_MATCHING_THRESHOLD
- NFMP_MAXIMAL_ROTATION
- NFMP_MODE
- NFMP_NAME
- NFMP_VERSION_HIGH
- NFMP_VERSION_LOW

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

See Also

NFMatcher Module | HNFMatcher | NfmSetParameter

6.5.1.5. NfmIdentifyEnd Function

Ends identification using the specified NFMatcher.

```
NResult N_API NfmIdentifyEnd(
        HNFMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to the NFMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* is NULL.
N_E_INVALID_OPERATION	Identification is not started.

See Also

NFMatcher Module | HNFMatcher | NfmIdentifyStart | NfmIdentifyNext

6.5.1.6. NfmIdentifyNext Function

Compares the specified packed NFRecord with the one identification was started with using the specified NFMatcher.

```
NResult N_API NfmIdentifyNext(
        HNFMatcher hMatcher,
        const void * templ,
        NSizeType templSize,
        NfmMatchDetails * pMatchDetails,
        NInt * pScore
);
```

Parameters

hMatcher	[in] Handle to the NFMatcher object.
templ	[in] Pointer to memory buffer containing packed NFRecord.
templSize	[in] Size of packed NFRecord.
pMatchDetails	[in, out] Pointer to NfmMatchDetails that is filled with details of the matching. Can be NULL.
pScore	[out] Pointer to NInt that receives similarity score.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher*, *templ*, or *pScore* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Identification is not started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see NFMP_MATCHING_THRESHOLD and NfmSetParameter function), and is greater than or equal to matching threshold otherwise.

If *pMatchDetails* is not NULL it should be a pointer obtained using NfmIdentifyStart function.

See Also

NFMatcher Module | HNFMatcher | NfmMatchDetails | NFMP_MATCHING_THRESHOLD | NfmSetParameter | NfmIdentifyStart | NfmIdentifyEnd

6.5.1.7. NfmIdentifyStart Function

Starts identification with the specified packed NFRecord using the specified NFMatcher.

```
NResult N_API NfmIdentifyStart(
        HNFMatcher hMatcher,
        const void * templ,
        NSizeType templSize,
        NfmMatchDetails * * ppMatchDetails
);
```

Parameters

hMatcher	[in] Handle to the NFMatcher object.
templ	[in] Pointer to memory buffer containing packed NFRecord.
templSize	[in] Size of packed NFRecord.

ppMatchDetails	[in] Pointer to pointer to NfmMatchDetails that receives pointer to created match details. Can be NULL.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* or *templ* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Identification is already started.

Remarks

If *ppMatchDetails* is not NULL, the received pointer should be susequently passed to NfmIdentifyNext function to obtain details of the matching.

Finally match details should be deleted using NfmMatchDetailsFree function.

See Also

NFMatcher Module | HNFMatcher | NfmMatchDetails | NfmMatchDetailsFree | NfmIdentifyNext | NfmIdentifyEnd

6.5.1.8. NfmIsRegistered Function

Checks if NFMatcher library is registered.

```
NBool N_API NfmIsRegistered(void);
```

Return Values

NTrue if library is registered, NFalse otherwise.

See Also

NFMatcher Module | NfmRegister | NfmUnregister

6.5.1.9. NfmMatchDetails Structure

Represents details of matching performed with NFMatcher.

```
typedef struct NfmMatchDetails_ { } NfmMatchDetails;
```

Fields

MatedMinutiaCount	Number of mated minutiae in first and second matched NFRecords.
MatedMinutiae	Pointer to array of NIndexPair containing pairs of indexes of mated minutiae in first and second matched NFRecords.
Score	Similarity score of two matched NFRecords.

See Also

NFMatcher Module

6.5.1.9.1. NfmMatchDetails.MatedMinutiaCount Field

Number of mated minutiae in first and second matched NFRecords.

```
NInt MatedMinutiaCount;
```

See Also

NfmMatchDetails

6.5.1.9.2. NfmMatchDetails.MatedMinutiae Field

Pointer to array of NIndexPair containing pairs of indexes of mated minutiae in first and second matched NFRecords.

```
NIndexPair * MatedMinutiae;
```

See Also

NfmMatchDetails | NIndexPair

6.5.1.9.3. NfmMatchDetailsScore.Score Field

Similarity score of two matched NFRecords.

```
NInt Score;
```

See Also

NfmMatchDetails

6.5.1.10. NfmMatchDetailsFree Function

Deletes NfmMatchDetails.

```
void N_API NfmMatchDetailsFree(
        NfmMatchDetails * pMatchDetails
);
```

Parameters

pMatchDetails	[in] Pointer to NfmMatchDetails to delete.

Remarks

If *pMatchDetails* is NULL, does nothing.

See Also

NFMatcher Module | NfmMatchDetails

6.5.1.11. NfmRegister Function

Registers NFMatcher library.

```
NResult N_API NfmRegister(
        const NChar * szServer
);
```

Parameters

szServer	[in] Pointer to string that specifies license manager server name or IP address.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*szServer* is NULL.
N_E_LM_CONNECTION	There was error during connection or communication with license manager server.
N_E_LM_NO_MORE_LICENSE	Registered license count achieved license manager license limit for NFMatcher library.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using `NfmUnregister` function.

See Also

NFMatcher Module | `NfmUnregister` | `NfmIsRegistered`

6.5.1.12. NfmReset Function

Sets default values for all parameters of the specified NFMatcher.

```
NResult N_API NfmReset(
        HNFMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to NFMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* is NULL.

See Also

NFMatcher Module | HNFMatcher

6.5.1.13. NfmSetParameter Function

Sets value of the specified parameter of the specified NFMatcher.

```
NResult N_API NfmSetParameter(
        HNFMatcher hMatcher,
        NUInt parameterId,
        const void * pValue
);
```

Parameters

hMatcher	[in] Handle to the NFMatcher object. Can be NULL if setting static parameter value.
parameterId	[in] Identifier of the parameter to set.
pValue	[in] Pointer to the parameter value to set.

Return Values

If the function succeeds, the return value is N_OK.

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

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

Copyright © 2006 Neurotechnologija

Remarks

The following values can be used for *parameterId*:

- NFMP_MATCHING_SPEED
- NFMP_MATCHING_THRESHOLD
- NFMP_MAXIMAL_ROTATION
- NFMP_MODE

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

See Also

NFMatcher Module | HNFMatcher | NfmGetParameter

6.5.1.14. NfmSpeed Enumeration

Specifies matching speed used in NFMatcher.

```
typedef enum NfmSpeed_ { } NfmSpeed;
```

Members

nfmsHigh	High matching speed (faster but less accurate).
nfmsLow	Low matching speed (slower but more accurate).
nfmsMedium	Medium matching speed (balance between speed and accuracy).

6.5.1.15. NfmUnregister Function

Unregisters NFMatcher library.

```
void N_API NfmUnregister(void);
```

See Also

NFMatcher Module | NfmRegister | NfmIsRegistered

6.5.1.16. NfmVerify Function

Compares two packed NFRecords using the specified NFMatcher.

```
NResult N_API NfmVerify(
        HNFMatcher hMatcher,
        const void * template1,
        NSizeType template1Size,
        const void * template2,
        NSizeType template2Size,
        NfmMatchDetails * * ppMatchDetails,
        NInt * pScore
);
```

Parameters

hMatcher	[in] Handle to the NFMatcher object.
template1	[in] Pointer to memory buffer containing first packed NFRecord.
template1Size	[in] Size of first packed NFRecord.
template2	[in] Pointer to memory buffer containing second packed NFRecord.
template2Size	[in] Size of second packed NFRecord.
ppMatchDetails	[in] Pointer to pointer to NfmMatchDetails that receives pointer to created match details. Can be NULL.
pScore	[out] Pointer to NInt that receives similarity score.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher*, *template1*, *template2*, or *pScore* is NULL.
N_E_END_OF_STREAM	*template1Size* or *template2Size* is less than expected.

Copyright © 2006 Neurotechnologija

Error Code	Condition
N_E_FORMAT	Data in memory buffer *template1* or *template2* points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Identification is started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see NFMP_MATCHING_THRESHOLD and NfmSetParameter function), and is greater than or equal to matching threshold otherwise.

If *ppMatchDetails* is not NULL, the received pointer should be examined to obtain details of the matching.

Finally match details should be deleted using NfmMatchDetailsFree function.

See Also

NFMatcher Module | HNFMatcher | NfmMatchDetails | NFMP_MATCHING_THRESHOLD | NfmSetParameter | NfmMatchDetailsFree | NfmIdentifyStart | NfmIdentifyNext | NfmIdentifyEnd

6.6. NFRecord Library

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

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

DLL (Windows): NFRecord.dll.

Shared object (Linux): libNFRecord.so.

Requirements (Windows):

- NCore.dll.

Requirements (Linux):

- libNCore.so.

Modules

NFRecord	Provides functionality for packing, unpack-

	ing and editing Neurotechnologija Finger Records (NFRecords).

6.6.1. NFRecord Module

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

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

Functions

`NFRecordAddCore`	Adds a NFCore to the end of NFRecord cores.
`NFRecordAddDelta`	Adds a NFDelta to the end of NFRecord deltas.
`NFRecordAddDoubleCore`	Adds a NFDoubleCore to the end of NFRecord double cores.
`NFRecordAddMinutia`	Adds a NFMinutia to the end of NFRecord minutiae.
`NFRecordCheck`	Checks if format of the packed NFRecord is correct.
`NFRecordClearCores`	Removes all cores from the NFRecord.
`NFRecordClearDeltas`	Removes all deltas from the NFRecord.
`NFRecordClearDoubleCores`	Removes all double cores from the NFRecord.
`NFRecordClearMinutiae`	Removes all minutiae from the NFRecord.
`NFRecordClone`	Creates a copy of the NFRecord.
`NFRecordCreate`	Creates an empty NFRecord.
`NFRecordCreateFromMemory`	Unpacks a NFRecord from the specified memory buffer.
`NFRecordFree`	Deletes the NFRecord. After the object is deleted the specified handle is no longer valid.
`NFRecordGetCore`	Retrieves the core at the specified index of the NFRecord.

Copyright © 2006 Neurotechnologija

NFRecordGetCoreCapacity	Retrieves the number of cores that the NFRecord can contain.
NFRecordGetCoreCount	Retrieves the number of cores in the NFRecord.
NFRecordGetCores	Copies all cores of NFRecord to the specified array.
NFRecordGetDelta	Retrieves the delta at the specified index of the NFRecord.
NFRecordGetDeltaCapacity	Retrieves the number of deltas that the NFRecord can contain.
NFRecordGetDeltaCount	Retrieves the number of deltas in the NFRecord.
NFRecordGetDeltas	Copies all deltas of NFRecord to the specified array.
NFRecordGetDoubleCore	Retrieves the double core at the specified index of the NFRecord.
NFRecordGetDoubleCoreCapacity	Retrieves the number of double cores that the NFRecord can contain.
NFRecordGetDoubleCoreCount	Retrieves the number of double cores in the NFRecord.
NFRecordGetDoubleCores	Copies all double cores of NFRecord to the specified array.
NFRecordGetG	Retrieves the G of the NFRecord.
NFRecordGetGMem	Retrieves the G of the packed NFRecord.
NFRecordGetHeight	Retrieves the height of the image the NFRecord is made from.
NFRecordGetHeightMem	Retrieves the height of the image the packed NFRecord is made from.
NFRecordGetHorzResolution	Retrieves the horizontal resolution of the image the NFRecord is made from.
NFRecordGetHorzResolutionMem	Retrieves the horizontal resolution of the image the packed NFRecord is made from.
NFRecordGetImpressionType	Retrieves the impression type of the NFRecord.
NFRecordGetImpressionTypeMem	Retrieves the impression type of the packed

Copyright © 2006 Neurotechnologija

	NFRecord.
NFRecordGetMaxSize	Retrieves the maximal size of a packed NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.
NFRecordGetMaxSizeV1	Retrieves the maximal size of a packed in version 1.0 format NFRecord containing the specified number of minutiae, cores, deltas and double cores and the specified ridge counts type.
NFRecordGetMinutia	Retrieves the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaCapacity	Retrieves the number of minutiae that the NFRecord can contain.
NFRecordGetMinutiaCount	Retrieves the number of minutiae in the NFRecord.
NFRecordGetMinutiaFormat	Retrieves the format of the minutiae in NFRecord.
NFRecordGetMinutiaNeighbour	Retrieves the minutia neighbour at the specified index of the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaNeighbourCount	Retrieves the number of minutia neighbours in the minutia at the specified index of the NFRecord.
NFRecordGetMinutiaNeighbours	Copies all minutia neighbours of the minutia at the specified index of the NFRecord to the specified array.
NFRecordGetMinutiae	Copies all minutiae of NFRecord to the specified array.
NFRecordGetPatternClass	Retrieves the pattern class of the NFRecord.
NFRecordGetPatternClassMem	Retrieves the pattern class of the packed NFRecord.
NFRecordGetPosition	Retrieves the finger position of the NFRecord.
NFRecordGetPositionMem	Retrieves the finger position of the packed NFRecord.
NFRecordGetQuality	Retrieves the quality of the NFRecord.

Copyright © 2006 Neurotechnologija

NFRecordGetQualityMem	Retrieves the quality of the packed NFRecord.
NFRecordGetRidgeCountsType	Retrieves the ridge counts type the NFRecord contains.
NFRecordGetSize	Calculates packed size of the NFRecord.
NFRecordGetSizeV1	Calculates packed in version 1.0 format size of the NFRecord.
NFRecordGetVertResolution	Retrieves the vertical resolution of the image the NFRecord is made from.
NFRecordGetVertResolutionMem	Retrieves the vertical resolution of the image the packed NFRecord is made from.
NFRecordGetWidth	Retrieves the width of the image the NFRecord is made from.
NFRecordGetWidthMem	Retrieves the width of the image the packed NFRecord is made from.
NFRecordInfoDispose	For internal use.
NFRecordInsertCore	Inserts a NFCore into the NFRecord at the specified index.
NFRecordInsertDelta	Inserts a NFDelta into the NFRecord at the specified index.
NFRecordInsertDoubleCore	Inserts a NFDoubleCore into the NFRecord at the specified index.
NFRecordInsertMinutia	Inserts a NFMinutia into the NFRecord at the specified index.
NFRecordRemoveCore	Removes the core at the specified index of the NFRecord.
NFRecordRemoveDelta	Removes the delta at the specified index of the NFRecord.
NFRecordRemoveDoubleCore	Removes the double core at the specified index of the NFRecord.
NFRecordRemoveMinutia	Removes the minutia at the specified index of the NFRecord.
NFRecordSaveToMemory	Packs the NFRecord into the specified memory buffer.
NFRecordSaveToMemoryV1	Packs the NFRecord into the specified

Copyright © 2006 Neurotechnologija

	memory buffer in version 1.0 format.
NFRecordSetCore	Sets a NFCore at the specified index of the NFRecord.
NFRecordSetCoreCapacity	Sets the number of cores that the NFRecord can contain.
NFRecordSetDelta	Sets a NFDelta at the specified index of the NFRecord.
NFRecordSetDeltaCapacity	Sets the number of deltas that the NFRecord can contain.
NFRecordSetDoubleCore	Sets a NFDoubleCore at the specified index of the NFRecord.
NFRecordSetDoubleCoreCapacity	Sets the number of double cores that the NFRecord can contain.
NFRecordSetG	Sets the G of the NFRecord.
NFRecordSetImpressionType	Sets the impression type of the NFRecord.
NFRecordSetMinutia	Sets a NFMinutia at the specified index of the NFRecord.
NFRecordSetMinutiaCapacity	Sets the number of minutiae that the NFRecord can contain.
NFRecordSetMinutiaNeighbour	Sets a NFMinutiaNeighbour at the specified index of the minutia at the specified index of the NFRecord.
NFRecordSetMinutiaFormat	Sets the format of the minutiae in NFRecord.
NFRecordSetPatternClass	Sets the pattern class of the NFRecord.
NFRecordSetPosition	Sets the finger position of the NFRecord.
NFRecordSetQuality	Sets the quality of the NFRecord.
NFRecordSetRidgeCountsType	Sets the ridge counts type the NFRecord contains.

Structures

NFCore	Represents a core in a NFRecord.
NFDelta	Represents a delta in a NFRecord.

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

Enumerations

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

Types

HNFRecord	Handle to NFRecord object.

Macros

NFR_BLOCK_SIZE	For internal use.
NFR_MAX_BLOCKED_ORIENTS_DIMENSION	For internal use.
NFR_MAX_CORE_COUNT	The maximum number of cores a NFRecord can contain.
NFR_MAX_DELTA_COUNT	The maximum number of deltas a NFRecord can contain.
NFR_MAX_DIMENSION	The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.
NFR_MAX_DOUBLE_CORE_COUNT	The maximum number of double cores a

Copyright © 2006 Neurotechnologija

	NFRecord can contain.
NFR_MAX_MINUTIA_COUNT	The maximum number of minutiae a NFRecord can contain.
NFR_NIST_RIDGE_COUNTS	For internal use.
NFR_RESOLUTION	The resolution of minutiae, cores, deltas and double cores coordinates in a NFRecord.
NFR_SAVE_BLOCKED_ORIENTS	The flag indicating whether blocked orientations should be packed in NFRecord.
NFR_SKIP_BLOCKED_ORIENTS	The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.
NFR_SKIP_CURVATURES	The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.
NFR_SKIP_GS	The flag indicating whether minutiae gs should be skipped while unpacking or packing NFRecord.
NFR_SKIP_QUALITIES	The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.
NFR_SKIP_RIDGE_COUNTS	The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.
NFR_SKIP_SINGULAR_POINTS	The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.

See Also

NFRecord Library

6.6.1.1. NFCore Structure

Represents a core in a NFRecord.

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

Fields

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

See Also

NFRecord Module

6.6.1.1.1. NFCore.Angle Field

Angle of this NFCore.

```
NInt Angle;
```

Remarks

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

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

See Also

NFCore

6.6.1.1.2. NFCore.X Field

X coordinate of this NFCore.

```
NUShort X;
```

Remarks

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

See Also

NFCore

6.6.1.1.3. NFCore.Y Field

Y coordinate of this NFCore.

```
NUShort Y;
```

Remarks

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

See Also

NFCore

6.6.1.2. NFDelta Structure

Represents a delta in a NFRecord.

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

Fields

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

See Also

NFRecord Module

6.6.1.2.1. NFDelta.Angle1 Field

First angle of this NFDelta.

```
NInt Angle1;
```

Remarks

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

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

See Also

NFDelta

6.6.1.2.2. NFDelta.Angle2 Field

Second angle of this NFDelta.

```
NInt Angle2;
```

Remarks

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

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

See Also

NFDelta

6.6.1.2.3. NFDelta.Angle3 Field

Third angle of this NFDelta.

```
NInt Angle3;
```

Remarks

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

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

See Also

NFDelta

6.6.1.2.4. NFDelta.X Field

X coordinate of this NFDelta.

```
NUShort X;
```

Remarks

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

See Also

NFDelta

6.6.1.2.5. NFDelta.Y Field

Y coordinate of this NFDelta.

```
NUShort Y;
```

Remarks

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

See Also

NFDelta

6.6.1.3. NFDoubleCore Structure

Represents a double core in a NFRecord.

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

Fields

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

See Also

NFRecord Module

6.6.1.3.1. NFDoubleCore.X Field

X coordinate of this NFDoubleCore.

```
NUShort X;
```

Remarks

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

See Also

NFDoubleCore

6.6.1.3.2. NFDoubleCore.Y Field

Y coordinate of this NFDoubleCore.

```
NUShort Y;
```

Remarks

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

See Also

NFDoubleCore

6.6.1.4. NFImpressionType Enumeration

Specifies the impression type.

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

Members

`nfitLatentImpression`	Latent impression fingerprint.
`nfitLatentLift`	Latent lift fingerprint.
`nfitLatentPhoto`	Latent photo fingerprint.
`nfitLatentTracing`	Latent tracing fingerprint.
`nfitLiveScanContactless`	Live-scanned fingerprint using contactless device.
`nfitLiveScanPlain`	Live-scanned plain fingerprint.
`nfitLiveScanRolled`	Live-scanned rolled fingerprint.

Copyright © 2006 Neurotechnologija

`nfitNonliveScanPlain`	Nonlive-scanned (from paper) plain fingerprint.
`nfitNonliveScanRolled`	Nonlive-scanned (from paper) rolled fingerprint.
`nfitSwipe`	Live-scanned fingerprint by sliding the finger across a "swipe" sensor.

Remarks

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

See Also

NFRecord Module

6.6.1.5. NFMinutia Structure

Represents a minutia in a NFRecord.

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

Fields

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

See Also

NFRecord Module

6.6.1.5.1. NFMinutia.Angle Field

Angle of this NFMinutia.

Copyright © 2006 Neurotechnologija

```
NByte Angle;
```

Remarks

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

See Also

NFMinutia

6.6.1.5.2. NFMinutia.Curvature Field

Ridge curvature near this NFMinutia.

```
NByte Curvature;
```

Remarks

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

See Also

NFMinutia

6.6.1.5.3. NFMinutia.G Field

G (ridge density) near this NFMinutia.

```
NByte G;
```

Remarks

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

See Also

NFMinutia

6.6.1.5.4. NFMinutia.Quality Field

Quality of this NFMinutia.

```
NByte Quality;
```

Remarks

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

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

See Also

NFMinutia

6.6.1.5.5. NFMinutia.Type Field

Type of this NFMinutia.

```
NFMinutiaType Type;
```

See Also

NFMinutia

6.6.1.5.6. NFMinutia.X Field

X coordinate of this NFMinutia.

```
NUShort X;
```

Remarks

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

See Also

NFMinutia

6.6.1.5.7. NFMinutia.Y Field

Y coordinate of this NFMinutia.

```
NUShort Y;
```

Remarks

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

See Also

NFMinutia

6.6.1.6. NFMinutiaFormat Enumeration

Specifies the minutia format.

This enumeration allows a bitwise combination of its member values.

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

Members

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

See Also

NFRecord Module | NFMinutia

6.6.1.7. NFMinutiaNeighbour Structure

Represents a minutia neighbour in a NFRecord.

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

Fields

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

See Also

NFRecord Module

6.6.1.7.1. NFMinutiaNeighbour.Index Field

Index of neighbour minutia.

```
NInt Index;
```

See Also

NFMinutiaNeighbour

6.6.1.7.2. NFMinutiaNeighbour.RidgeCount Field

Ridge count to neighbour minutia.

```
NByte RidgeCount;
```

See Also

NFMinutiaNeighbour

6.6.1.8. NFMinutiaType Enumeration

Specifies the minutia type.

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

Members

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

See Also

NFRecord Module | NFMinutia

6.6.1.9. NFPatternClass Enumeration

Specifies the pattern class of the fingerprint.

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

Members

nfpcAccidentalWhorl	Accidental whorl pattern class.
nfpcAmputation	Amputation. Pattern class is not available.
nfpcCentralPocketLoop	Central pocket loop pattern class.
nfpcDoubleLoop	Double loop pattern class.
nfpcLeftSlantLoop	Left slant loop pattern class.
nfpcPlainArch	Plain arch pattern class.
nfpcPlainWhorl	Plain whorl pattern class.
nfpcRadialLoop	Radial loop pattern class.
nfpcRightSlantLoop	Right slant loop pattern class.
nfpcScar	Scar. Pattern class is not available.
nfpcTentedArch	Tented arch pattern class.
nfpcUlnarLoop	Ulnar loop pattern class.
nfpcUnknown	Unknown pattern class.
nfpcWhorl	Whorl pattern class.

Remarks

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

See Also

NFRecord Module

6.6.1.10. NFPosition Enumeration

Specifies the finger position.

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

Members

nfpLeftIndex	Index finger of the left hand.
nfpLeftLittle	Little finger of the left hand.
nfpLeftMiddle	Middle finger of the left hand.

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

Remarks

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

See Also

NFRecord Module

6.6.1.11. NFRecordAddCore Function

Adds a NFCore to the end of NFRecord cores.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFCore

6.6.1.12. NFRecordAddDelta Function

Adds a NFDelta to the end of NFRecord deltas.

```
NResult N_API NFRecordAddDelta(
        HNFRecord hRecord,
        const NFDelta * pValue
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFDelta

6.6.1.13. NFRecordAddDoubleCore Function

Adds a NFDoubleCore to the end of NFRecord double cores.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFDoubleCore

6.6.1.14. NFRecordAddMinutia Function

Adds a NFMinutia to the end of NFRecord minutiae.

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

```
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFMinutia

6.6.1.15. NFRecordCheck Function

Checks if format of the packed NFRecord is correct.

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

Parameters

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

If the function succeeds, the return value is N_OK.

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

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

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

See Also

NFRecord Module

6.6.1.16. NFRecordClearCores Function

Removes all cores from the NFRecord.

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

Parameters

hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord

6.6.1.17. NFRecordClearDeltas Function

Removes all deltas from the NFRecord.

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

Parameters

hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord

6.6.1.18. NFRecordClearDoubleCores Function

Removes all double cores from the NFRecord.

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

Parameters

hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord

6.6.1.19. NFRecordClearMinutiae Function

Removes all minutiae from the NFRecord.

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

Parameters

hRecord	[in] Handle to the NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

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

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord

6.6.1.20. NFRecordClone Function

Creates a copy of the NFRecord.

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

```
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Created object must be deleted using `NFRecordFree` function.

See Also

NFRecord Module | HNFRecord | `NFRecordFree`

6.6.1.21. NFRecordCreate Function

Creates an empty NFRecord.

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

Parameters

width	[in] Specifies width of fingerprint image.

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

The following flags are supported:

- NFR_NIST_RIDGE_COUNTS

Created object must be deleted using NFRecordFree function.

See Also

NFRecord Module | HNFRecord | NFRecordFree

6.6.1.22. NFRecordCreateFromMemory Function

Unpacks a NFRecord from the specified memory buffer.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

The following flags are supported:

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

This function supports both NFRecord version 1.0 and 2.0 formats.

Created object must be deleted using `NFRecordFree` function.

See Also

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

6.6.1.23. NFRecordFree Function

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

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

Parameters

hRecord	[in] Handle to the NFRecord object.

Remarks

If *hRecord* is NULL, does nothing.

See Also

NFRecord Module | HNFRecord

6.6.1.24. NFRecordGetCore Function

Retrieves the core at the specified index of the NFRecord.

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

```
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFCore | NFRecordGetCoreCount | NFRecordSetCore

6.6.1.25. NFRecordGetCoreCapacity Function

Retrieves the number of cores that the NFRecord can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

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

6.6.1.26. NFRecordGetCoreCount Function

Retrieves the number of cores in the NFRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

NFRecord Module | HNFRecord | NFRecordGetCoreCapacity | NFRecordSetCoreCapacity

6.6.1.27. NFRecordGetCores Function

Copies all cores of NFRecord to the specified array.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

NFRecord Module | HNFRecord | NFCore | NFRecordGetCoreCount

6.6.1.28. NFRecordGetDelta Function

Retrieves the delta at the specified index of the NFRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFDelta | NFRecordGetDeltaCount | NFRecordSetDelta

Copyright © 2006 Neurotechnologija

6.6.1.29. NFRecordGetDeltaCapacity Function

Retrieves the number of deltas that the NFRecord can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

NFRecord Module | HNFRecord | `NFRecordSetDeltaCapacity` | `NFRecordGetDeltaCount`

6.6.1.30. NFRecordGetDeltaCount Function

Retrieves the number of deltas in the NFRecord.

```
NResult N_API NFRecordGetDeltaCount(
        HNFRecord hRecord,
```

Copyright © 2006 Neurotechnologija

```
        NInt * pValue
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

NFRecord Module | HNFRecord | `NFRecordGetDeltaCapacity` | `NFRecordSetDeltaCapacity`

6.6.1.31. NFRecordGetDeltas Function

Copies all deltas of NFRecord to the specified array.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

NFRecord Module | HNFRecord | NFDelta | NFRecordGetDeltaCount

6.6.1.32. NFRecordGetDoubleCore Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFDoubleCore | NFRecordGetDoubleCoreCount | NFRecordSetDoubleCore

6.6.1.33. NFRecordGetDoubleCoreCapacity Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

NFRecord Module | HNFRecord | `NFRecordSetDoubleCoreCapacity` | `NFRecordGetDoubleCoreCount`

6.6.1.34. NFRecordGetDoubleCoreCount Function

Retrieves the number of double cores in the NFRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

Copyright © 2006 Neurotechnologija

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

See Also

NFRecord Module | HNFRecord | `NFRecordGetDoubleCoreCapacity` | `NFRecord-SetDoubleCoreCapacity`

6.6.1.35. NFRecordGetDoubleCores Function

Copies all double cores of NFRecord to the specified array.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

See Also

NFRecord Module | HNFRecord | NFDoubleCore | `NFRecordGetDoubleCoreCount`

6.6.1.36. NFRecordGetG Function

Retrieves the G of the NFRecord.

Copyright © 2006 Neurotechnologija

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFRecordSetG

6.6.1.37. NFRecordGetGMem Function

Retrieves the G of the packed NFRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

See Also

NFRecord Module

6.6.1.38. NFRecordGetHeight Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord

6.6.1.39. NFRecordGetHeightMem Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `1` for version 1.0 format.

See Also

NFRecord Module

6.6.1.40. NFRecordGetHorzResolution Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord

6.6.1.41. NFRecordGetHorzResolutionMem Function

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

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `500` for version 1.0 format.

See Also

NFRecord Module

6.6.1.42. NFRecordGetImpressionType Function

Retrieves the impression type of the NFRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFImpressionType | NFRecordSetImpressionType

6.6.1.43. NFRecordGetImpressionTypeMem Function

Retrieves the impression type of the packed NFRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `nfitLiveScanPlain` for version 1.0 format.

See Also

NFRecord Module | NFImpressionType

6.6.1.44. NFRecordGetMaxSize Function

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

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

Parameters

minutiaFormat	[in] The minutia format.
minutiaCount	[in] The number of minutiae.
ridgeCountsType	[in] The type of ridge counts.
coreCount	[in] The number of cores.
deltaCount	[in] The number of deltas.
doubleCoreCount	[in] The number of double cores.

boWidth	[in] The width of blocked orientations.
boHeight	[in] The height of blocked orientations.
pSize	[out] Pointer to NSizeType that receives maximal size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

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

Error Code	Condition
	NFR_MAX_BLOCKED_ORIENTS_DIMENSION.

Remarks

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

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

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

See Also

NFRecord Module | NFMinutiaFormat | NFMinutia | NFRidgeCountsType | NFCore | NFDelta | NFDoubleCore | NFRecordSaveToMemory

6.6.1.45. NFRecordGetMaxSizeV1 Function

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

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

Parameters

minutiaFormat	[in] The minutia format.
minutiaCount	[in] The number of minutiae.
coreCount	[in] The number of cores.
deltaCount	[in] The number of deltas.
doubleCoreCount	[in] The number of double cores.
boWidth	[in] The width of blocked orientations.

boHeight	[in] The height of blocked orientations.
pSize	[out] Pointer to NSizeType that receives maximal size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

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

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

See Also

NFRecord Module | NFMinutiaFormat | NFMinutia | NFCore | NFDelta | NFDoubleCore | NFRecordSaveToMemoryV1

6.6.1.46. NFRecordGetMinutia Function

Retrieves the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordGetMinutia(
        HNFRecord hRecord,
        NInt index,
        NFMinutia * pValue
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFMinutia | NFRecordGetMinutiaCount | NFRe-

cordGetMinutiae

6.6.1.47. NFRecordGetMinutiaCapacity Function

Retrieves the number of minutiae that the NFRecord can contain.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Minutia capacity is the number of minutiae that the NFRecord can store. Minutia count (see NFRecordGetMinutiaCount function) is the number of minutiae that are actually in the NFRecord.

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

See Also

NFRecord Module | HNFRecord | NFRecordSetMinutiaCapacity | NFRecordGet-MinutiaCount

6.6.1.48. NFRecordGetMinutiaCount Function

Retrieves the number of minutiae in the NFRecord.

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

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

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

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

Remarks

Minutia capacity (see `NFRecordGetMinutiaCapacity` and `NFRecordSetMinutiaCapacity` functions) is the number of minutiae that the NFRecord can store. Minutia count is the number of minutiae that are actually in the NFRecord.

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

See Also

NFRecord Module | HNFRecord | `NFRecordGetMinutiaCapacity` | `NFRecordSetMinutiaCapacity`

6.6.1.49. NFRecordGetMinutiaFormat Function

Retrieves the format of the minutiae in NFRecord.

```
NResult N_API NFRecordGetMinutiaFormat(
        HNFRecord hRecord,
        NFMinutiaFormat * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFMinutiaFormat that receives format of minutiae in the NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

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

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

See Also

NFRecord Module | HNFRecord | NFMinutiaFormat | NFRecordSetMinutiaFormat

6.6.1.50. NFRecordGetMinutiaNeighbour Function

Retrieves the minutia neighbour at the specified index of the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordGetMinutiaNeighbour(
        HNFRecord hRecord,
        NInt minutiaIndex,
        NInt index,
        NFMinutiaNeighbour * pValue
);
```

Parameters

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

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaIndex* is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function. - or - *index* is less than zero or greater than or equal to minutia neighbour count obtained using NFRecordGetMinutiaNeighbourCount function.

See Also

NFRecord Module | HNFRecord | NFMinutiaNeighbour | NFRecordGetMinutiaCount | NFRecordGetMinutiaNeighbourCount | NFRecordSetMinutiaNeighbour

6.6.1.51. NFRecordGetMinutiaNeighbourCount Function

Retrieves the number of minutia neighbours in the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordGetMinutiaNeighbourCount(
        HNFRecord hRecord,
        NInt minutiaIndex,
        NInt * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
minutiaIndex	[in] The index of minutia.
pValue	[out] Pointer to NInt that receives number of minutia neighbours.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

NFRecord Module | HNFRecord

6.6.1.52. NFRecordGetMinutiaNeighbours Function

Copies all minutia neighbours of the minutia at the specified index of the NFRecord to the specified array.

```
NResult N_API NFRecordGetMinutiaNeighbours(
        HNFRecord hRecord,
        NInt minutiaIndex,
        NFMinutiaNeighbour * arMinutiaNeighbours
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
minutiaIndex	[in] The index of minutia.
arMinutiaNeighbours	[out] Pointer to array of NFMinutiaNeighbour that receives minutia neighbours.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *arMinutiaNeighbours* is NULL.

Remarks

Array *arMinutiaNeighbours* points to must be large enough to receive all minutia neighbours. See NFRecordGetMinutiaNeighbourCount function.

See Also

NFRecord Module | HNFRecord | NFMinutiaNeighbour | NFRecordGetMinutiaCount | NFRecordGetMinutiaNeighbourCount

6.6.1.53. NFRecordGetMinutiae Function

Copies all minutiae of NFRecord to the specified array.

```
NResult N_API NFRecordGetMinutiae(
        HNFRecord hRecord,
        NFMinutia * arMinutiae
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
arMinutiae	[out] Pointer to array of NFMinutiathat receives minutiae.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *arMinutiae*is NULL.

Remarks

Array *arMinutiae* points to must be large enough to receive all NFRecord minutiae. See NFRecordGetMinutiaCount function.

See Also

NFRecord Module | HNFRecord | NFMinutia | NFRecordGetMinutiaCount

6.6.1.54. NFRecordGetPatternClass Function

Retrieves the pattern class of the NFRecord.

```
NResult N_API NFRecordGetPatternClass(
        HNFRecord hRecord,
        NFPatternClass * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFPatternClass that receives fingerprint pattern class.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | NFPatternClass | NFRecordSetPatternClass

6.6.1.55. NFRecordGetPatternClassMem Function

Retrieves the pattern class of the packed NFRecord.

```
NResult N_API NFRecordGetPatternClassMem(
        const void * buffer,
        NSizeType bufferSize,
        NFPatternClass * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.

pValue	[out] Pointer to NFPatternClass that receives fingerprint pattern class.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `nfpcUnknown` for version 1.0 format.

See Also

NFRecord Module | NFPatternClass

6.6.1.56. NFRecordGetPosition Function

Retrieves the finger position of the NFRecord.

```
NResult N_API NFRecordGetPosition(
        HNFRecord hRecord,
        NFPosition * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFPosition that receives finger position.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | NFPosition | NFRecordSetPosition

6.6.1.57. NFRecordGetPositionMem Function

Retrieves the finger position of the packed NFRecord.

```
NResult N_API NFRecordGetPositionMem(
        const void * buffer,
        NSizeType bufferSize,
        NFPosition * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NFPosition that receives finger position.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Copyright © 2006 Neurotechnologija

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `nfpUnknown` for version 1.0 format.

See Also

NFRecord Module | NFPosition

6.6.1.58. NFRecordGetQuality Function

Retrieves the quality of the NFRecord.

```
NResult N_API NFRecordGetQuality(
        HNFRecord hRecord,
        NByte * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NByte that receives fingerprint quality.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | `NFRecordSetQuality`

6.6.1.59. NFRecordGetQualityMem Function

Retrieves the quality of the packed NFRecord.

```
NResult N_API NFRecordGetQualityMem(
        const void * buffer,
```

Copyright © 2006 Neurotechnologija

```
        NSizeType bufferSize,
        NByte * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NByte that receives finger-print quality.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `0` for version 1.0 format.

See Also

NFRecord Module

6.6.1.60. NFRecordGetRidgeCountsType Function

Retrieves the ridge counts type the NFRecord contains.

```
NResult N_API NFRecordGetRidgeCountsType(
        HNFRecord hRecord,
        NFRidgeCountsType * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NFRidgeCountsType that receives ridge counts type stored in NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord | NFRidgeCountsType | NFRecordSetRidgeCountsType

6.6.1.61. NFRecordGetSize Function

Calculates packed size of the NFRecord.

```
NResult N_API NFRecordGetSize(
        HNFRecord hRecord,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFRecord.

Return Values

Copyright © 2006 Neurotechnologija

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL.

Remarks

The function calculates current (2.0) version packed size of NFRecord.

For the list of flags that are supported see `NFRecordSaveToMemory` function.

See Also

NFRecord Module | HNFRecord | `NFRecordSaveToMemory`

6.6.1.62. NFRecordGetSizeV1 Function

Calculates packed in version 1.0 format size of the NFRecord.

```
NResult N_API NFRecordGetSizeV1(
        HNFRecord hRecord,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL.

Copyright © 2006 Neurotechnologija

Remarks

For the list of flags that are supported see NFRecordSaveToMemoryV1 function.

See Also

NFRecord Module | HNFRecord | NFRecordSaveToMemoryV1

6.6.1.63. NFRecordGetVertResolution Function

Retrieves the vertical resolution of the image the NFRecord is made from.

```
NResult N_API NFRecordGetVertResolution(
        HNFRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NUShort that receives vertical resolution of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord

6.6.1.64. NFRecordGetVertResolutionMem Function

Retrieves the vertical resolution of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetVertResolutionMem(
        const void * buffer,
        NSizeType bufferSize,
        NUShort * pValue
);
```

Copyright © 2006 Neurotechnologija

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NUShort that receives vertical resolution of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `500` for version 1.0 format.

See Also

NFRecord Module

6.6.1.65. NFRecordGetWidth Function

Retrieves the width of the image the NFRecord is made from.

```
NResult N_API NFRecordGetWidth(
        HNFRecord hRecord,
        NUShort * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
pValue	[out] Pointer to NUShort that receives width of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

See Also

NFRecord Module | HNFRecord

6.6.1.66. NFRecordGetWidthMem Function

Retrieves the width of the image the packed NFRecord is made from.

```
NResult N_API NFRecordGetWidthMem(
        const void * buffer,
        NSizeType bufferSize,
        NUShort * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
pValue	[out] Pointer to NUShort that receives width of fingerprint image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.

Remarks

This function supports both NFRecord version 1.0 and 2.0 formats.

Always returns `1` for version 1.0 format.

See Also

NFRecord Module

6.6.1.67. NFRecordInsertCore Function

Inserts a NFCore into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertCore(
        HNFRecord hRecord,
        NInt index,
        const NFCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index at which core is inserted.
pValue	[in] Pointer to the NFCore to insert.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than core count obtained using NFRecordGetCoreCount function.
N_E_INVALID_OPERATION	Number of cores in NFRecord (see NFRecordGetCoreCount) is equal to NFR_MAX_CORE_COUNT.

See Also

NFRecord Module | HNFRecord | NFCore | NFRecordGetCoreCount

6.6.1.68. NFRecordInsertDelta Function

Inserts a NFDelta into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertDelta(
        HNFRecord hRecord,
        NInt index,
        const NFDelta * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index at which delta is inserted.
pValue	[in] Pointer to the NFDelta to insert.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than delta

Error Code	Condition
	count obtained using NFRecordGetDeltaCount function.
N_E_INVALID_OPERATION	Number of deltas in NFRecord (see NFRecordGetDeltaCount) is equal to NFR_MAX_DELTA_COUNT.

See Also

NFRecord Module | HNFRecord | NFDelta | NFRecordGetDeltaCount

6.6.1.69. NFRecordInsertDoubleCore Function

Inserts a NFDoubleCore into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertDoubleCore(
        HNFRecord hRecord,
        NInt index,
        const NFDoubleCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index at which double core is inserted.
pValue	[in] Pointer to the NFDoubleCore to insert.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than double core count obtained using NFRecordGetDoubleCoreCount function.

Error Code	Condition
N_E_INVALID_OPERATION	Number of double core in NFRecord (see NFRecordGetDoubleCoreCount) is equal to NFR_MAX_DOUBLE_CORE_COUNT.

See Also

NFRecord Module | HNFRecord | NFDoubleCore | NFRecordGetDoubleCoreCount

6.6.1.70. NFRecordInsertMinutia Function

Inserts a NFMinutia into the NFRecord at the specified index.

```
NResult N_API NFRecordInsertMinutia(
        HNFRecord hRecord,
        NInt index,
        const NFMinutia * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index at which minutia is inserted.
pValue	[in] Pointer to the NFMinutia to insert.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than minutia count obtained using NFRecord-GetMinutiaCount function.
N_E_INVALID_OPERATION	Number of minutia in NFRecord (see NFRecordGetMinutiaCount) is equal

Error Code	Condition
	to NFR_MAX_MINUTIA_COUNT.

See Also

NFRecord Module | HNFRecord | NFMinutia | `NFRecordGetMinutiaCount`

6.6.1.71. NFRecordRemoveCore Function

Removes the core at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveCore(
        HNFRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of core to remove.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to core count obtained using `NFRecordGetCoreCount` function.

See Also

NFRecord Module | HNFRecord | `NFRecordGetCoreCount`

6.6.1.72. NFRecordRemoveDelta Function

Removes the delta at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveDelta(
        HNFRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of delta to remove.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount function.

See Also

NFRecord Module | HNFRecord | NFRecordGetDeltaCount

6.6.1.73. NFRecordRemoveDoubleCore Function

Removes the double core at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveDoubleCore(
        HNFRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of double core to remove.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to double core count obtained using NFRecordGetDoubleCoreCount function.

See Also

NFRecord Module | HNFRecord | NFRecordGetDoubleCoreCount

6.6.1.74. NFRecordRemoveMinutia Function

Removes the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordRemoveMinutia(
        HNFRecord hRecord,
        NInt index
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of minutia to remove.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

NFRecord Module | HNFRecord | `NFRecordGetMinutiaCount`

6.6.1.75. NFRecordSaveToMemory Function

Packs the NFRecord into the specified memory buffer.

```
NResult N_API NFRecordSaveToMemory(
        HNFRecord hRecord,
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
buffer	[out] Pointer to memory buffer to store packed NFRecord. Can be NULL.
bufferSize	[in] Size of memory buffer to store packed NFRecord.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store packed NFRecord.
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not

Error Code	Condition
	zero.

Remarks

The function packs NFRecord in current (2.0) version format.

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as NFRecordGetSize function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with NFRecordGetSize function.

Note that blocked orientations are not packed by default.

The following flags are supported:

- NFR_SAVE_BLOCKED_ORIENTS
- NFR_SKIP_CURVATURES
- NFR_SKIP_GS
- NFR_SKIP_QUALITIES
- NFR_SKIP_RIDGE_COUNTS
- NFR_SKIP_SINGULAR_POINTS

See Also

NFRecord Module | HNFRecord | NFRecordGetSize | NFRecordCreateFromMemory

6.6.1.76. NFRecordSaveToMemoryV1 Function

Packs the NFRecord into the specified memory buffer in version 1.0 format.

```
NResult N_API NFRecordSaveToMemoryV1(
        HNFRecord hRecord,
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
buffer	[out] Pointer to memory buffer to store

	packed NFRecord. Can be NULL.
bufferSize	[in] Size of memory buffer to store packed NFRecord.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFRecord.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store packed NFRecord.
N_E_ARGUMENT_NULL	*hRecord* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not zero.

Remarks

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as `NFRecordGetSizeV1` function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with `NFRecordGetSizeV1` function.

Note that blocked orientations are not packed by default.

The following flags are supported:

- NFR_SAVE_BLOCKED_ORIENTS
- NFR_SKIP_CURVATURES
- NFR_SKIP_GS
- NFR_SKIP_SINGULAR_POINTS

Copyright © 2006 Neurotechnologija

See Also

NFRecord Module | HNFRecord | NFRecordCreateFromMemory | NFRecordGetSizeV1

6.6.1.77. NFRecordSetCore Function

Sets a NFCore at the specified index of the NFRecord.

```
NResult N_API NFRecordSetCore(
        HNFRecord hRecord,
        NInt index,
        const NFCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of core to set.
pValue	[in] Pointer to the NFCore to set.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to core count obtained using NFRecordGetCoreCount function.

See Also

NFRecord Module | HNFRecord | NFCore | NFRecordGetCoreCount | NFRecordGetCore

6.6.1.78. NFRecordSetCoreCapacity Function

Copyright © 2006 Neurotechnologija

Sets the number of cores that the NFRecord can contain.

```
NResult N_API NFRecordSetCoreCapacity(
        HNFRecord hRecord,
        NInt value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New number of cores NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than core count obtained using `NFRecordGetCoreCount` function.

Remarks

Core capacity is the number of cores that the NFRecord can store. Core count (see `NFRecordGetCoreCount` function) is the number of cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | `NFRecordGetCoreCapacity` | `NFRecordGetCoreCount`

6.6.1.79. NFRecordSetDelta Function

Sets a NFDelta at the specified index of the NFRecord.

```
NResult N_API NFRecordSetDelta(
        HNFRecord hRecord,
```

```
        NInt index,
        const NFDelta * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of delta to set.
pValue	[in] Pointer to the NFDelta to set.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to delta count obtained using NFRecordGetDeltaCount function.

See Also

NFRecord Module | HNFRecord | NFDelta | NFRecordGetDeltaCount | NFRecordGetDelta

6.6.1.80. NFRecordSetDeltaCapacity Function

Sets the number of deltas that the NFRecord can contain.

```
NResult N_API NFRecordSetDeltaCapacity(
        HNFRecord hRecord,
        NInt value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New number of deltas NFRecord can

	contain.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than delta count obtained using `NFRecordGetDeltaCount` function.

Remarks

Delta capacity is the number of deltas that the NFRecord can store. Delta count (see `NFRecordGetDeltaCount` function) is the number of deltas that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding deltas the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | `NFRecordGetDeltaCapacity` | `NFRecordGetDeltaCount`

6.6.1.81. NFRecordSetDoubleCore Function

Sets a NFDoubleCore at the specified index of the NFRecord.

```
NResult N_API NFRecordSetDoubleCore(
        HNFRecord hRecord,
        NInt index,
        const NFDoubleCore * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of double core to set.
pValue	[in] Pointer to the NFDoubleCore to set.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to double core count obtained using NFRecordGetDoubleCoreCount function.

See Also

NFRecord Module | HNFRecord | NFDoubleCore | NFRecordGetDoubleCoreCount | NFRecordGetDoubleCore

6.6.1.82. NFRecordSetDoubleCoreCapacity Function

Sets the number of double cores that the NFRecord can contain.

```
NResult N_API NFRecordSetDoubleCoreCapacity(
        HNFRecord hRecord,
        NInt value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New number of double cores NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than double core count obtained using NFRecordGetDoubleCoreCount function.

Remarks

Double core capacity is the number of double cores that the NFRecord can store. Double core count (see NFRecordGetDoubleCoreCount function) is the number of double cores that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding double cores the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordGetDoubleCoreCapacity | NFRecordGetDoubleCoreCount

6.6.1.83. NFRecordSetG Function

Sets the G of the NFRecord.

```
NResult N_API NFRecordSetG(
        HNFRecord hRecord,
        NByte value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New G value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFRecordGetG

6.6.1.84. NFRecordSetImpressionType Function

Sets the impression type of the NFRecord.

```
NResult N_API NFRecordSetImpressionType(
        HNFRecord hRecord,
        NFImpressionType value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New impression type value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFImpressionType | NFRecordGetImpressionType

6.6.1.85. NFRecordSetMinutia Function

Sets a NFMinutia at the specified index of the NFRecord.

```
NResult N_API NFRecordSetMinutia(
        HNFRecord hRecord,
        NInt index,
        const NFMinutia * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
index	[in] Index of minutia to set.
pValue	[in] Pointer to the NFMinutia to set.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function.

See Also

NFRecord Module | HNFRecord | NFMinutia | NFRecordGetMinutiaCount | NFRecordGetMinutia

6.6.1.86. NFRecordSetMinutiaCapacity Function

Sets the number of minutiae that the NFRecord can contain.

```
NResult N_API NFRecordSetMinutiaCapacity(
        HNFRecord hRecord,
        NInt value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New number of minutiae NFRecord can contain.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than minutia count obtained using NFRecordGetMinutiaCount function.

Remarks

Minutia capacity is the number of minutiae that the NFRecord can store. Minutia count (see NFRecordGetMinutiaCount function) is the number of minutiae that are actually in the NFRecord.

Capacity is always greater than or equal to count. If count exceeds capacity while adding minutiae the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFRecord Module | HNFRecord | NFRecordGetMinutiaCapacity | NFRecordGetMinutiaCount

6.6.1.87. NFRecordSetMinutiaFormat Function

Sets the format of the minutiae in NFRecord.

```
NResult N_API NFRecordSetMinutiaFormat(
        HNFRecord hRecord,
        NFMinutiaFormat value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New minutia format value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFMinutiaFormat | NFRecordGetMinutiaFormat

6.6.1.88. NFRecordSetMinutiaNeighbour Function

Sets a NFMinutiaNeighbour at the specified index of the minutia at the specified index of the NFRecord.

```
NResult N_API NFRecordSetMinutiaNeighbour(
        HNFRecord hRecord,
        NInt minutiaIndex,
        NInt index,
        const NFMinutiaNeighbour * pValue
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
minutiaIndex	[in] The index of minutia.
index	[in] Index of minutia neighbour to set.
pValue	[in] Pointer to the NFMinutiaNeighbour to set.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*hRecord* or *pValue* is NULL.

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	*minutiaIndex* is less than zero or greater than or equal to minutia count obtained using NFRecordGetMinutiaCount function. - or - *index* is less than zero or greater than or equal to minutia neighbour count obtained using NFRecordGetMinutiaNeighbourCount function.

See Also

NFRecord Module | HNFRecord | NFMinutiaNeighbour | NFRecordGetMinutiaCount | NFRecordGetMinutiaNeighbourCount | NFRecordGetMinutiaNeighbour

6.6.1.89. NFRecordSetPatternClass Function

Sets the pattern class of the NFRecord.

```
NResult N_API NFRecordSetPatternClass(
        HNFRecord hRecord,
        NFPatternClass value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New fingerprint pattern class value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFPatternClass | `NFRecordGetPatternClass`

6.6.1.90. NFRecordSetPosition Function

Sets the finger position of the NFRecord.

```
NResult N_API NFRecordSetPosition(
        HNFRecord hRecord,
        NFPosition value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New finger position value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFPosition | `NFRecordGetPosition`

6.6.1.91. NFRecordSetQuality Function

Sets the quality of the NFRecord.

```
NResult N_API NFRecordSetQuality(
        HNFRecord hRecord,
        NByte value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New fingerprint quality value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFRecordGetQuality

6.6.1.92. NFRecordSetRidgeCountsType Function

Sets the ridge counts type the NFRecord contains.

```
NResult N_API NFRecordSetRidgeCountsType(
        HNFRecord hRecord,
        NFRidgeCountsType value
);
```

Parameters

hRecord	[in] Handle to the NFRecord object.
value	[in] New ridge counts type value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*value* is invalid.
N_E_ARGUMENT_NULL	*hRecord* is NULL.

See Also

NFRecord Module | HNFRecord | NFRidgeCountsType | `NFRecordGetRidgeCountsType`

6.6.1.93. NFRidgeCountsType Enumeration

Specifies the type of ridge counts contained in a NFRecord.

```
typedef enum NFRidgeCountsType_ { } NFRidgeCountsType;
```

Members

`nfrctEightNeighbours`	The NFRecord contains ridge counts to closest minutia in each of the eight sectors of each minutia. First sector starts at minutia angle.
`nfrctEightNeighboursWithIndexes`	The NFRecord contains ridge counts to eight neighbours of each minutia.
`nfrctFourNeighbours`	The NFRecord contains ridge counts to closest minutia in each of the four sectors of each minutia. First sector starts at minutia angle.
`nfrctFourNeighboursWithIndexes`	The NFRecord contains ridge counts to four neighbours of each minutia.
`nfrctNone`	The NFRecord does not contain ridge counts.
`nfrctUnspecified`	For internal use.

See Also

NFRecord Module

6.7. NfsMatcher Library

Provides functionality for comparing Neurotechnologija Fingers Templates.

Import library (Windows): `NfsMatcher.dll.lib`.

DLL (Windows): `NfsMatcher.dll`.

Shared object (Linux): `libNfsMatcher.so`.

Requirements (Windows):

- `NCore.dll`.
- `NFMatcher.dll`.
- `NFRecord.dll`.
- `NFTemplate.dll`.
- `VFMatcher.dll`.

Requirements (Linux):

- `libNCore.so`.
- `libNFMatcher.so`.
- `libNFRecord.so`.
- `libNFTemplate.so`.
- `libVFMatcher.so`.

Modules

NfsMatcher	Provides functionality for comparing Neurotechnologija Fingers Templates (NFTemplates), encapsulated in Neurotechnologija Fingers Matcher (NfsMatcher) object.

6.7.1. NfsMatcher Module

Provides functionality for comparing Neurotechnologija Fingers Templates (NFTemplates), encapsulated in Neurotechnologija Fingers Matcher (NfsMatcher) object.

Header file: `NfsMatcher.h` (includes `NfsMatcherParams.h` and `NfsmMatchDetails.h`).

Functions

`NfsmCopyParameters`	Copies parameter values from one NfsMatcher to another.
`NfsmCreate`	Creates a NfsMatcher.
`NfsmFree`	Deletes the NfsMatcher. After the object is deleted the specified handle is no longer valid.
`NfsmGetParameter`	Retrieves value of the specified parameter of the specified NfsMatcher.

Copyright © 2006 Neurotechnologija

NfsmIdentifyEnd	Ends identification using the specified Nfs-Matcher.
NfsmIdentifyNext	Compares the specified packed NFTemplate with the one identification was started with using the specified NfsMatcher.
NfsmIdentifyStart	Starts identification with the specified packed NFTemplate using the specified Nfs-Matcher.
NfsmIsRegistered	Checks if NfsMatcher library is registered.
NfsmMatchDetailsFree	Deletes NfsmMatchDetailsFree.
NfsmRegister	Registers NfsMatcher library.
NfsmReset	Sets default values for all parameters of the specified NfsMatcher.
NfsmSetParameter	Sets value of the specified parameter of the specified NfsMatcher.
NfsmUnregister	Unregisters NfsMatcher library.
NfsmVerify	Compares two packed NFTemplate using the specified NfsMatcher.

Structures

NfsmFingerMatchDetails	Represents details of matching performed with NfsMatcher for one finger in the first matched NFTemplate.
NfsmMatchDetails	Represents details of matching performed with NfsMatcher.

Enumerations

NfsmUsedMatcher	Specifies matcher used for finger matching in NfsMatcher.

Types

HNfsMatcher	Handle to NfsMatcher object.

Macros

NFSM_PART_NFM	Identifier specifying that retrieving or setting parameter for NFMatcher contained in NfsMatcher.
NFSM_PART_NONE	Identifier specifying that retrieving or setting parameter for NfsMatcher.
NFSM_PART_VFM	Identifier specifying that retrieving or setting parameter for VFMatcher contained in NfsMatcher.
NFSMP_COPYRIGHT	Identifier specifying library copyright static read-only parameter of type N_TYPE_STRING.
NFSMP_MATCHING_THRESHOLD	Identifier specifying matching threshold (biggest allowed FAR) parameter of type N_TYPE_INT. Parameter value is equal to -12 * log10(FAR) and must be not less than zero (for example, 48 for FAR = 0.01%).
NFSMP_NAME	Identifier specifying library name static read-only parameter of type N_TYPE_STRING.
NFSMP_USED_MATCHER	Specifies used matcher. One of the NfsmUsedMatcher values.
NFSMP_VERSION_HIGH	Identifier specifying high part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
NFSMP_VERSION_LOW	Identifier specifying low part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

See Also

NfsMatcher Library

6.7.1.1. NfsmCopyParameters Function

Copies parameter values from one NfsMatcher to another.

```
NResult N_API NfsmCopyParameters(
        HNfsMatcher hDstMatcher,
        HNfsMatcher hSrcMatcher
);
```

Parameters

hDstMatcher	[in] Handle to the destination NfsMatcher object.
hSrcMatcher	[in] Handle to the source NfsMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hDstMatcher* or *hSrcMatcher* is NULL.
N_E_INVALID_OPERATION	Identification is started on *hDstMatcher*.

See Also

NfsMatcher Module | HNfsMatcher

6.7.1.2. NfsmCreate Function

Creates a NfsMatcher.

```
NResult N_API NfsmCreate(
        HNfsMatcher * pHMatcher
);
```

Parameters

pHMatcher	[out] Pointer to HNfsMatcher that receives handle to created NfsMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pHMatcher* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NfsmFree` function.

See Also

NfsMatcher Module | HNfsMatcher | `NfsmFree`

6.7.1.3. NfsmFingerMatchDetails Structure

Represents details of matching performed with NfsMatcher for one finger in the first matched NFTemplate.

```
typedef struct NfsmFingerMatchDetails_ { } NfsmFingerMatchDetails;
```

Fields

MatchedIndex	Index of matched finger in the second matched NFTemplate.
NfmMatchDetails	Pointer to NfmMatchDetails containing matching details from NFMatcher.
Score	Similarity score of two matched fingers.
VfmMatchDetails	Pointer to VfmMatchDetails containing matching details from VFMatcher.

See Also

NfsMatcher Module

6.7.1.3.1. NfsmFingerMatchDetails.MatchedIndex Field

Index of matched finger in the second matched NFTemplate.

```
NInt MatchedIndex;
```

Remarks

The field has value of -1 if no match found.

See Also

NfsmFingerMatchDetails

6.7.1.3.2. NfsmFingerMatchDetails.NfmMatchDetails Field

Pointer to NfmMatchDetails containing matching details from NFMatcher.

```
NfmMatchDetails * NfmMatchDetails;
```

Remarks

The field has value of NULL if NFMatcher was not used.

See Also

NfsmFingerMatchDetails | NfmMatchDetails

6.7.1.3.3. NfsmFingerMatchDetails.Score Field

Similarity score of two matched fingers.

```
NInt Score;
```

See Also

NfsmFingerMatchDetails

6.7.1.3.4. NfsmFingerMatchDetails.VfmMatchDetails Field

Pointer to VfmMatchDetails containing matching details from VFMatcher.

```
VfmMatchDetails * VfmMatchDetails;
```

Remarks

The field has value of NULL if VFMatcher was not used.

See Also

NfsmFingerMatchDetails | VfmMatchDetails

6.7.1.4. NfsmFree Function

Deletes the NfsMatcher. After the object is deleted the specified handle is no longer valid.

```
void N_API NfsmFree(
        HNfsMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to NfsMatcher object.

Remarks

If *hMatcher* is NULL, does nothing.

See Also

NfsMatcher Module | HNfsMatcher

6.7.1.5. NfsmGetParameter Function

Retrieves value of the specified parameter of the specified NfsMatcher.

```
NResult N_API NfsmGetParameter(
        HNfsMatcher hMatcher,
        NUShort partId,
        NUInt parameterId,
        void * pValue
);
```

Parameters

hMatcher	[in] Handle to the NfsMatcher object. Can be NULL if retrieving static parameter value.
partId	[in] Identifier of the part of the matcher.
parameterId	[in] Identifier of the parameter to retrieve.
pValue	[out] Pointer to variable that receives parameter value.

Copyright © 2006 Neurotechnologija

Return Values

If the function succeeds and *parameterId* specifies a N_TYPE_STRING type parameter, and *pValue* is NULL, the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*partId* is invalid.
N_E_ARGUMENT_NULL	*parameterId* specifies a non-static parameter and *hMatcher* is NULL. - or - *parameterId* specifies a non-N_TYPE_STRING type parameter and *pValue* is NULL.
N_E_PARAMETER	*parameterId* is invalid.

Remarks

partId can be one of the following:

- NFSM_PART_NFM
- NFSM_PART_NONE
- NFSM_PART_VFM

If *partId* is NFSM_PART_NFM, *parameterId* must be any `NfmGetParameter` function supports except for NFMP_MATCHING_THRESHOLD.

If *partId* is NFSM_PART_VFM, *parameterId* must be any `VfmGetParameter` function supports except for VFMP_MATCHING_THRESHOLD.

If *partId* is NFSM_PART_NONE, the following values can be used for *parameterId*:

- NFSMP_COPYRIGHT
- NFSMP_MATCHING_THRESHOLD
- NFSMP_NAME
- NFSMP_USED_MATCHER
- NFSMP_VERSION_HIGH
- NFSMP_VERSION_LOW

To learn the type of the parameter pass value obtained with NParameterMakeId macro using N_PC_TYPE_ID code and the parameter id via *parameterId* parameter and pointer to NInt that will receive one of N_TYPE_XXX via *pValue* parameter. *hMatcher* can be NULL in this case.

See Also

NfsMatcher Module | HNfsMatcher | NfmGetParameter | VfmGetParameter | NfsmSetParameter

6.7.1.6. NfsmIdentifyEnd Function

Ends identification using the specified NfsMatcher.

```
NResult N_API NfsmIdentifyEnd(
        HNfsMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to the NfsMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* is NULL.
N_E_INVALID_OPERATION	Identification is not started.

See Also

NfsMatcher Module | HNfsMatcher | NfsmIdentifyStart | NfsmIdentifyNext

6.7.1.7. NfsmIdentifyNext Function

Compares the specified packed NFTemplate with the one identification was started with using the specified NfsMatcher.

```
NResult N_API NfsmIdentifyNext(
        HNfsMatcher hMatcher,
        const void * templ,
```

```
        NSizeType templSize,
        NfsmMatchDetails * pMatchDetails,
        NInt * pScore
);
```

Parameters

hMatcher	[in] Handle to the NfsMatcher object.
templ	[in] Pointer to memory buffer containing packed NFTemplate.
templSize	[in] Size of packed NFTemplate.
pMatchDetails	[in, out] Pointer to NfsmMatchDetails that is filled with details of the matching. Can be NULL.
pScore	[out] Pointer to NInt that receives similarity score.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher*, *templ*, or *pScore* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NFTemplate format.
N_E_INVALID_OPERATION	Identification is not started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see NFSMP_MATCHING_THRESHOLD and NfsmSetParameter function), and is greater than or equal to matching threshold otherwise.

If *pMatchDetails* is not NULL it should be a pointer obtained using NfsmIdentifyStart function.

See Also

Copyright © 2006 Neurotechnologija

NfsMatcher Module | HNfsMatcher | NfsmMatchDetails | NFSMP_MATCHING_THRESHOLD | `NfsmSetParameter` | `NfsmIdentifyStart` | `NfsmIdentifyEnd`

6.7.1.8. NfsmIdentifyStart Function

Starts identification with the specified packed NFTemplate using the specified NfsMatcher.

```
NResult N_API NfsmIdentifyStart(
        HNfsMatcher hMatcher,
        const void * templ,
        NSizeType templSize,
        NfsmMatchDetails * * ppMatchDetails
);
```

Parameters

hMatcher	[in] Handle to the NfsMatcher object.
templ	[in] Pointer to memory buffer containing packed NFTemplate.
templSize	[in] Size of packed NFTemplate.
ppMatchDetails	[in] Pointer to pointer to NfsmMatchDetails that receives pointer to created match details. Can be NULL.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* or *templ* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NFTemplate format.
N_E_INVALID_OPERATION	Identification is already started.

Remarks

Copyright © 2006 Neurotechnologija

If *ppMatchDetails* is not NULL, the received pointer should be susequently passed to NfsmIdentifyNext function to obtain details of the matching.

Finally match details should be deleted using NfsmMatchDetailsFree function.

See Also

NfsMatcher Module | HNfsMatcher | NfsmMatchDetails | NfsmMatchDetailsFree | NfsmIdentifyNext | NfsmIdentifyEnd

6.7.1.9. NfsmIsRegistered Function

Checks if NfsMatcher library is registered.

```
NBool N_API NfsmIsRegistered(void);
```

Return Values

NTrue if library is registered, NFalse otherwise.

See Also

NfsMatcher Module | NfsmRegister | NfsmUnregister

6.7.1.10. NfsmMatchDetails Structure

Represents details of matching performed with NfsMatcher.

```
typedef struct NfsmMatchDetails_ { } NfsmMatchDetails;
```

Fields

FingerCount	Number of fingers in the first matched NFTemplate.
Fingers	Pointer to array of NfsmFingerMatchDetails containing details of matching performed for each finger in the first matched NFTemplate.
Score	Similarity score of two matched NFTemplates.

See Also

NfsMatcher Module

6.7.1.10.1. NfsmMatchDetails.FingerCount Field

Number of fingers in the first matched NFTemplate.

```
NInt FingerCount;
```

See Also

NfsmMatchDetails

6.7.1.10.2. NfsmMatchDetails.Fingers Field

Pointer to array of NfsmFingerMatchDetails containing details of matching performed for each finger in the first matched NFTemplate.

```
NfsmFingerMatchDetails * Fingers;
```

See Also

NfsmMatchDetails | NfsmFingerMatchDetails

6.7.1.10.3. NfsmMatchDetails.Score Field

Similarity score of two matched NFTemplates.

```
NInt Score;
```

See Also

NfsmMatchDetails

6.7.1.11. NfsmMatchDetailsFree Function

Deletes NfsmMatchDetailsFree.

```
void N_API NfsmMatchDetailsFree(
        NfsmMatchDetails * pMatchDetails
);
```

Parameters

pMatchDetails	[in] Pointer to NfsmMatchDetails to delete.

Remarks

If *pMatchDetails* is NULL, does nothing.

See Also

NfsMatcher Module | NfsmMatchDetails

6.7.1.12. NfsmRegister Function

Registers NfsMatcher library.

```
NResult N_API NfsmRegister(
        const NChar * szServer
);
```

Parameters

szServer	[in] Pointer to string that specifies license manager server name or IP address.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szServer* is NULL.
N_E_LM_CONNECTION	There was error during connection or communication with license manager server.
N_E_LM_NO_MORE_LICENSE	Registered license count achieved license manager license limit for NfsMatcher library.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using `NfsmUnregister` function.

See Also

NfsMatcher Module | `NfsmUnregister` | `NfsmIsRegistered`

6.7.1.13. NfsmReset Function

Copyright © 2006 Neurotechnologija

Sets default values for all parameters of the specified NfsMatcher.

```
NResult N_API NfsmReset(
        HNfsMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to NfsMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* is NULL.

See Also

NfsMatcher Module | HNfsMatcher

6.7.1.14. NfsmSetParameter Function

Sets value of the specified parameter of the specified NfsMatcher.

```
NResult N_API NfsmSetParameter(
        HNfsMatcher hMatcher,
        NUShort partId,
        NUInt parameterId,
        const void * pValue
);
```

Parameters

hMatcher	[in] Handle to the NfsMatcher object. Can be NULL if setting static parameter value.
partId	[in] Identifier of the part of the matcher.
parameterId	[in] Identifier of the parameter to set.
pValue	[in] Pointer to the parameter value to set.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*partId* is invalid. - or - Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*parameterId* specifies a non-static parameter and *hMatcher* is NULL. - or - *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	Value *pValue* points to is out of range.
N_E_INVALID_OPERATION	*hMatcher* is not NULL and identification is started on the matcher.
N_E_PARAMETER	*parameterId* is invalid.
N_E_PARAMETER_READ_ONLY	*parameterId* specifies read-only parameter.

Remarks

partId can be one of the following:

- NFSM_PART_NFM
- NFSM_PART_NONE
- NFSM_PART_VFM

If *partId* is NFSM_PART_NFM, *parameterId* must be any `NfmSetParameter` function supports except for NFMP_MATCHING_THRESHOLD.

If *partId* is NFSM_PART_VFM, *parameterId* must be any `VfmSetParameter` function supports except for VFMP_MATCHING_THRESHOLD.

If *partId* is NFSM_PART_NONE, the following values can be used for *parameterId*:

- NFSMP_MATCHING_THRESHOLD

Copyright © 2006 Neurotechnologija

- NFSMP_USED_MATCHER

To learn the type of the parameter pass value obtained with NParameterMakeId macro using N_PC_TYPE_ID code and the parameter id via *parameterId* parameter and pointer to NInt that will receive one of N_TYPE_XXX via *pValue* parameter. *hMatcher* can be NULL in this case.

See Also

NfsMatcher Module | HNfsMatcher | NfmSetParameter | VfmSetParameter | NfsmGetParameter

6.7.1.15. NfsmUnregister Function

Unregisters NfsMatcher library.

```
void N_API NfsmUnregister(void);
```

See Also

NfsMatcher Module | NfsmRegister | NfsmIsRegistered

6.7.1.16. NfsmUsedMatcher Enumeration

Specifies matcher used for finger matching in NfsMatcher.

```
typedef enum NfsmUsedMatcher_ { } NfsmUsedMatcher;
```

Members

nfsmumNF	NFMatcher is used.
nfsmumNFAndVFFusion	NFMatcher and VFMatcher fusion is used.
nfsmumVF	VFMatcher is used.

6.7.1.17. NfsmVerify Function

Compares two packed NFTemplate using the specified NfsMatcher.

```
NResult N_API NfsmVerify(
        HNfsMatcher hMatcher,
        const void * template1,
        NSizeType template1Size,
        const void * template2,
        NSizeType template2Size,
        NfsmMatchDetails * * ppMatchDetails,
        NInt * pScore
```

```
);
```

Parameters

hMatcher	[in] Handle to the NfsMatcher object.
template1	[in] Pointer to memory buffer containing first packed NFTemplate.
template1Size	[in] Size of first packed NFTemplate.
template2	[in] Pointer to memory buffer containing second packed NFTemplate.
template2Size	[in] Size of second packed NFTemplate.
ppMatchDetails	[in] Pointer to pointer to NfsmMatchDetails that receives pointer to created match details. Can be NULL.
pScore	[out] Pointer to NInt that receives similarity score.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher*, *template1*, *template2*, or *pScore* is NULL.
N_E_END_OF_STREAM	*template1Size* or *template2Size* is less than expected.
N_E_FORMAT	Data in memory buffer *template1* or *template2* points to is inconsistent with NFTemplate format.
N_E_INVALID_OPERATION	Identification is started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see NFSMP_MATCHING_THRESHOLD and NfsmSetParameter function), and is greater than or equal to matching threshold otherwise.

Copyright © 2006 Neurotechnologija

If *ppMatchDetails* is not NULL, the received pointer should be examined to obtain details of the matching.

Finally match details should be deleted using NfsmMatchDetailsFree function.

See Also

NfsMatcher Module | HNfsMatcher | NfsmMatchDetails | NFSMP_MATCHING_THRESHOLD | NfsmSetParameter | NfsmMatchDetailsFree | NfsmIdentifyStart | NfsmIdentifyNext | NfsmIdentifyEnd

6.8. NFTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates.

Import library (Windows): NFTemplate.dll.lib.

DLL (Windows): NFTemplate.dll.

Shared object (Linux): libNFTemplate.so.

Requirements (Windows):

- NCore.dll.
- NFRecord.dll.

Requirements (Linux):

- libNCore.so.
- libNFRecord.so.

Modules

NFTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).

6.8.1. NFTemplate Module

Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).

Header file: NFTemplate.h

Functions

NFTemplateAddRecord	Adds an empty finger record to the NFTemplate.
NFTemplateAddRecordCopy	Adds a copy of the finger record to the NFTemplate.
NFTemplateAddRecordFromMemory	Unpacks a finger record from the specified memory buffer and adds it to the NFTemplate.
NFTemplateCalculateSize	Calculates the size of a packed NFTemplate containing the specified number of finger records of specified size.
NFTemplateCheck	Checks if format of the packed NFTemplate is correct.
NFTemplateClearRecords	Removes all finger records from the NFTemplate.
NFTemplateClone	Creates a copy of the NFTemplate.
NFTemplateCreate	Creates an empty NFTemplate.
NFTemplateCreateFromMemory	Unpacks a NFTemplate from the specified memory buffer.
NFTemplateFree	Deletes the NFTemplate. After the object is deleted the specified handle is no longer valid.
NFTemplateGetRecord	Retrieves the finger record at the specified index of the NFTemplate.
NFTemplateGetRecordCapacity	Retrieves the number of finger records that the NFTemplate can contain..
NFTemplateGetRecordCount	Retrieves the number of finger records in the NFTemplate.
NFTemplateGetRecordCountMem	Retrieves the number of finger records in the packed NFTemplate.
NFTemplateGetSize	Calculates packed size of the NFTemplate.
NFTemplateInfoDispose	For internal use.
NFTemplatePack	Packs packed finger records as NFTemplate into the specified memory buffer.

Copyright © 2006 Neurotechnologija

NFTemplateRemoveRecord	Removes the finger record at the specified index of the NFTemplate.
NFTemplateSaveToMemory	Packs the NFTemplate into the specified memory buffer.
NFTemplateSetRecordCapacity	Sets the number of finger records that the NFTemplate can contain.
NFTemplateUnpack	Unpacks packed finger records from the packed NFTemplate.

Structures

NFTemplateInfo	For internal use.

Types

HNFTemplate	Handle to NFTemplate object.

Macros

NFT_MAX_RECORD_COUNT	The maximum number of finger records NFTemplate can contain.
NFT_PROCESS_FIRST_RECORD_ONLY	The flag indicating whether only the first finger record should be unpacked or packed while unpacking or packing NFTemplate.

See Also

NFTemplate Library

6.8.1.1. NFTemplateAddRecord Function

Adds an empty finger record to the NFTemplate.

```
NResult N_API NFTemplateAddRecord(
        HNFTemplate hTemplate,
        NUShort width,
        NUShort height,
        NUShort horzResolution,
        NUShort vertResolution,
```

```
        NUInt flags,
        HNFRecord * pHRecord
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
width	[in] Specifies width of fingerprint image.
height	[in] Specifies height of fingerprint image.
horzResolution	[in] Specifies horizontal resolution of fingerprint image.
vertResolution	[in] Specifies vertical resolution of fingerprint image.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is zero.
N_E_ARGUMENT_NULL	*hTemplate* or *pHRecord* is NULL.
N_E_INVALID_OPERATION	Number of finger records in NFTemplate (see NFTemplateGetRecordCount) is equal to NFT_MAX_RECORD_COUNT.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see `NFRecordCreate` function.

See Also

NFTemplate Module | HNFTemplate | HNFRecord | `NFTemplateGetRecordCount` | `NFRecordCreate`

6.8.1.2. NFTemplateAddRecordCopy Function

Adds a copy of the finger record to the NFTemplate.

```
NResult N_API NFTemplateAddRecordCopy(
        HNFTemplate hTemplate,
        HNFRecord hSrcRecord,
        HNFRecord * pHRecord
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
hSrcRecord	[in] Handle to the NFRecord object.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	**Condition**
N_E_ARGUMENT_NULL	*hTemplate* or *hSrcRecord* or *pHRecord* is NULL.
N_E_INVALID_OPERATION	Number of finger records in NFTemplate (see `NFTemplateGetRecordCount`) is equal to NFT_MAX_RECORD_COUNT.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

NFTemplate Module | HNFTemplate | HNFRecord | `NFTemplateGetRecordCount`

6.8.1.3. NFTemplateAddRecordFromMemory Function

Unpacks a finger record from the specified memory buffer and adds it to the NFTemplate.

```
NResult N_API NFTemplateAddRecordFromMemory(
        HNFTemplate hTemplate,
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        HNFRecord * pHRecord
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
buffer	[in] Pointer to memory buffer that contains packed NFRecord.
bufferSize	[in] Size of memory buffer that contains packed NFRecord.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHRecord*, or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Number of finger records in NFTemplate (see `NFTemplateGetRecordCount`) is equal to NFT_MAX_RECORD_COUNT.

Copyright © 2006 Neurotechnologija

Error Code	Condition
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see `NFRecordCreateFromMemory` function.

See Also

NFTemplate Module | HNFTemplate | HNFRecord | `NFTemplateGetRecordCount` | `NFRecordCreateFromMemory`

6.8.1.4. NFTemplateCalculateSize Function

Calculates the size of a packed NFTemplate containing the specified number of finger records of specified size.

```
NResult N_API NFTemplateCalculateSize(
        NInt recordCount,
        NSizeType * arRecordSizes,
        NSizeType * pSize
);
```

Parameters

recordCount	[in] The number of finger records.
arRecordSizes	[in] Pointer to array of NSizeType that contains packed sizes of each finger record.
pSize	[out] Pointer to NSizeType that receives calculated size of packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Any entry of the array *arRecordSizes* points to is less than minimal finger record size.
N_E_ARGUMENT_NULL	*arRecordSizes* or *pSize* is NULL.

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	*recordCount* is less than zero or greater than NFT_MAX_RECORD_COUNT.

Remarks

This is a low-level function and can be changed in future version of the library.

Use NFRecordGetMaxSize function to calculate packed size of an individual finger record.

See Also

NFTemplate Module | NFRecordGetMaxSize | NFTemplateSaveToMemory

6.8.1.5. NFTemplateCheck Function

Checks if format of the packed NFTemplate is correct.

```
NResult N_API NFTemplateCheck(
        const void * buffer,
        NSizeType bufferSize
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer*buffer* points to is inconsistent with NFTemplate format.

Copyright © 2006 Neurotechnologija

See Also

NFTemplate Module

6.8.1.6. NFTemplateClearRecords Function

Removes all finger records from the NFTemplate.

```
NResult N_API NFTemplateClearRecords(
        HNFTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.

See Also

NFTemplate Module | HNFTemplate

6.8.1.7. NFTemplateClone Function

Creates a copy of the NFTemplate.

```
NResult N_API NFTemplateClone(
        HNFTemplate hTemplate,
        HNFTemplate * pHClonedTemplate
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
pHClonedTemplate	[out] Pointer to HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHClonedTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using NFTemplateFree function.

See Also

NFTemplate Module | HNFTemplate | NFTemplateFree

6.8.1.8. NFTemplateCreate Function

Creates an empty NFTemplate.

```
NResult N_API NFTemplateCreate(
        HNFTemplate * pHTemplate
);
```

Parameters

pHTemplate	[out] Pointer to HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pHTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NFTemplateFree` function.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateFree`

6.8.1.9. NFTemplateCreateFromMemory Function

Unpacks a NFTemplate from the specified memory buffer.

```
NResult N_API NFTemplateCreateFromMemory(
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NFTemplateInfo * pInfo,
        HNFTemplate * pHTemplate
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pInfo	[out] For internal use. Must be NULL.
pHTemplate	[out] Pointer to HNFTemplate that receives handle to newly created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pHTemplate* or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is

Error Code	Condition
	inconsistent with NFTemplate format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

The following flags are supported:

- NFT_PROCESS_FIRST_RECORD_ONLY
- Flags supported by `NFRecordCreateFromMemory` function are applied to each finger record contained in the NFTemplate.

Created object must be deleted using `NFTemplateFree` function.

See Also

NFTemplate Module | HNFTemplate | NFTemplateInfo | `NFTemplateFree` | `NFRecordCreateFromMemory` | `NFTemplateSaveToMemory`

6.8.1.10. NFTemplateFree Function

Deletes the NFTemplate. After the object is deleted the specified handle is no longer valid.

```
void N_API NFTemplateFree(
        HNFTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.

Remarks

If *hTemplate* is NULL, does nothing.

See Also

NFTemplate Module | HNFTemplate

6.8.1.11. NFTemplateGetRecord Function

Retrieves the finger record at the specified index of the NFTemplate.

```
NResult N_API NFTemplateGetRecord(
        HNFTemplate hTemplate,
        NInt index,
        HNFRecord * pValue
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
index	[in] Index of finger record to retrieve.
pValue	[out] Pointer to HNFRecord that receives handle to NFRecord object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to finger record count obtained using `NFTemplateGetRecordCount` function.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateGetRecordCount`

Copyright © 2006 Neurotechnologija

6.8.1.12. NFTemplateGetRecordCapacity Function

Retrieves the number of finger records that the NFTemplate can contain..

```
NResult N_API NFTemplateGetRecordCapacity(
        HNFTemplate hTemplate,
        NInt * pValue
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
pValue	[out] Pointer to NInt that receives number of finger records NFTemplate can contain.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pValue* is NULL.

Remarks

Finger record capacity is the number of finger records that the NFTemplate can store. Finger record count (see `NFTemplateGetRecordCount` function) is the number of finger records that are actually in the NFTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateSetRecordCapacity` | `NFTemplateGetRecordCount`

6.8.1.13. NFTemplateGetRecordCount Function

Retrieves the number of finger records in the NFTemplate.

```
NResult N_API NFTemplateGetRecordCount(
```

```
        HNFTemplate hTemplate,
        NInt * pValue
);
```

Parameters

hTemplate	[in] Handle to NFTemplate object.
pValue	[out] Pointer to NInt that receives number of finger records.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pValue* is NULL.

Remarks

Finger record capacity (see `NFTemplateGetRecordCapacity` and `NFTemplateSetRecordCapacity` functions) is the number of finger records that the NFTemplate can store. Finger record count is the number of finger records that are actually in the NFTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateGetRecordCapacity` | `NFTemplateSetRecordCapacity`

6.8.1.14. NFTemplateGetRecordCountMem Function

Retrieves the number of finger records in the packed NFTemplate.

```
NResult N_API NFTemplateGetRecordCountMem(
        const void * buffer,
        NSizeType bufferSize,
        NInt * pValue
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
pValue	[out] Pointer to NInt that receives number of finger records.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* or *pValue* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.

See Also

NFTemplate Module

6.8.1.15. NFTemplateGetSize Function

Calculates packed size of the NFTemplate.

```
NResult N_API NFTemplateGetSize(
        HNFTemplate hTemplate,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hTemplate	[in] Handle to NFTemplate object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.

pSize	[out] Pointer to NSizeType that receives calculated size of packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pSize* is NULL.

Remarks

For the list of flags that are supported see `NFTemplateSaveToMemory` function.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateSaveToMemory`

6.8.1.16. NFTemplatePack Function

Packs packed finger records as NFTemplate into the specified memory buffer.

```
NResult N_API NFTemplatePack(
        NInt recordCount,
        const void * * arRecords,
        NSizeType * arRecordSizes,
        void * buffer,
        NSizeType bufferSize,
        NSizeType * pSize
);
```

Parameters

recordCount	[in] The number of finger records.
arRecords	[in] Pointer to array of void * that contains packed finger records.
arRecordSizes	[in] Pointer to array of NSizeType that contains packed sizes of each finger record.
buffer	[out] Pointer to memory buffer to store packed NFTemplate.

bufferSize	[in] Size of memory buffer to store packed NFTemplate.
pSize	[out] Pointer to NSizeType that receives size of packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*bufferSize* is less than size required to store packed NFTemplate. - or - Any entry of the array *arRecordSizes* points to is less than minimal finger record size. - or - Any entry of the array *arRecords* points to is NULL. - or - Any entry of the array *arRecordSizes* points to is not equal to size stored in memory buffer accroding entry of the array *arRecords* points to, points to.
N_E_ARGUMENT_NULL	*arRecords* or *buffer* or *pSize* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*recordCount* is less than zero or greater than NFT_MAX_RECORD_COUNT.
N_E_FORMAT	Data in memory buffer any entry of the array *arRecords* points to is inconsistent with NFRecord format.

Remarks

This is a low-level function and can be changed in future version of the library.

bufferSize must not be less than value calculated with NFTemplateCalculateSize function with the same parameter values.

See Also

NFTemplate Module | NFTemplateCalculateSize | NFTemplateUnpack

6.8.1.17. NFTemplateRemoveRecord Function

Removes the finger record at the specified index of the NFTemplate.

```
NResult N_API NFTemplateRemoveRecord(
        HNFTemplate hTemplate,
        NInt index
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
index	[in] Index of finger record to remove.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater or equal than records count.

See Also

NFTemplate Module | HNFTemplate | NFTemplateGetRecordCount

6.8.1.18. NFTemplateSaveToMemory Function

Packs the NFTemplate into the specified memory buffer.

```
NResult N_API NFTemplateSaveToMemoryEx(
        HNFTemplate hTemplate,
```

```
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hTemplate	[in] Handle to NFTemplate object.
buffer	[out] Pointer to memory buffer to store packed NFTemplate. Can be NULL.
bufferSize	[in] Size of memory buffer to store packed NFTemplate.
flags	[in] Bitwise combination of zero or more flags that control behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NFTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store packed NFTemplate.
N_E_ARGUMENT_NULL	*hTemplate* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not zero.

Remarks

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as `NFTemplateGetSize` function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with `NFTemplateGetSize` function.

Copyright © 2006 Neurotechnologija

The following flags are supported:

- NFT_PROCESS_FIRST_RECORD_ONLY
- Flags supported by `NFRecordSaveToMemory` function are applied to each finger record contained in the NFTemplate.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateGetSize` | `NFRecordSaveToMemory` | `NFTemplateCreateFromMemory`

6.8.1.19. NFTemplateSetRecordCapacity Function

Sets the number of finger records that the NFTemplate can contain.

```
NResult N_API NFTemplateSetRecordCapacity(
        HNFTemplate hTemplate,
        NInt value
);
```

Parameters

hTemplate	[in] Handle to the NFTemplate object.
value	[in] New number of finger records NFTemplate can contain.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*value* is less than finger record count obtained using `NFTemplateGetRecordCount` function.

Remarks

Finger record capacity is the number of finger records that the NFTemplate can store. Finger record count (see `NFTemplateGetRecordCount` function) is the number of finger re-

cords that are actually in the NFTemplate.

Capacity is always greater than or equal to count. If count exceeds capacity while adding finger records the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

See Also

NFTemplate Module | HNFTemplate | `NFTemplateGetRecordCapacity` | `NFTemplateGetRecordCount`

6.8.1.20. NFTemplateUnpack Function

Unpacks packed finger records from the packed NFTemplate.

```
NResult N_API NFTemplateUnpack(
        const void * buffer,
        NSizeType bufferSize,
        NByte * pMajorVersion,
        NByte * pMinorVersion,
        NUInt * pSize,
        NByte * pHeaderSize,
        NInt * pRecordCount,
        const void * * arRecords,
        NSizeType * arRecordSizes
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
pMajorVersion	[out] Pointer to NByte that receives major version of the packed NFTemplate. Can be NULL.
pMinorVersion	[out] Pointer to NByte that receives minor version of the packed NFTemplate. Can be NULL.
pSize	[out] Pointer to NUInt that receives size of the packed NFTemplate. Can be NULL.
pHeaderSize	[out] Pointer to NByte that receives header size of the packed NFTemplate. Can be NULL.
pRecordCount	[out] Pointer to NInt that receives number of finger records contained in the packed

	NFTemplate. Can be NULL.
arRecords	[out] Pointer to array of void * that receives pointers to packed finger records contained in the packed NFTemplate. Can be NULL.
arRecordSizes	[out] Pointer to array of NSizeType that receives sizes of packed finger records contained in the packed NFTemplate. Can be NULL.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.

Remarks

This is a low-level function and can be changed in future version of the library.

This function should be first called with *arRecords* and *arRecordSizes* parameters set to NULL to receive finger record count through *pRecordCount* parameter. Then arrays of finger records count length should be allocated and this function should be called with theese array passed in *arRecords* and *arRecordSizes* parameters.

See Also

NFTemplate Module | NFTemplatePack

6.9. NImages Pro Library

Provides functionality for loading, saving and converting images in various formats.

Import library (Windows): NImagesPro.dll.lib.

DLL (Windows): NImagesPro.dll.

Shared object (Linux): `libNImagesPro.so`.

Requirements (Windows):

- `NCore.dll`.

Requirements (Linux):

- `libNCore.so`.

Modules

Bmp	Provides functionality for loading and saving images in BMP format.
IHead	Provides functionality for loading and saving images in NIST IHead format.
NGrayscaleImage	Provides functionality for managing 8-bit grayscale images.
NImageFile	Provides functionality for reading image files in format-neutral way.
NImageFormat Pro	Provides functionality for loading and saving images in format-neutral way.
NImage Pro	Provides functionality for managing images.
NImages Pro	Provides library registration and other additional functionality.
NMonochromeImage	Provides functionality for managing 1-bit monochrome images.
NPixelFormat	Provides functionality for work with image pixel format.
NRgbImage	Provides functionality for managing 24-bit RGB images.
Tiff	Provides functionality for loading images in TIFF format.
Wsq	Provides functionality for loading and saving images in WSQ format.

Copyright © 2006 Neurotechnologija

6.9.1. Bmp Module

Provides functionality for loading and saving images in BMP format.

Header file: `Bmp.h`.

Functions

`BmpLoadImageFromFile`	Loads image from BMP file.
`BmpLoadImageFromHBitmap`	Loads image from Windows HBITMAP.
`BmpLoadImageFromMemory`	Loads image from memory buffer containing BMP file.
`BmpSaveImageToFile`	Saves image to file in BMP format.
`BmpSaveImageToHBitmap`	Saves image to Windows HBITMAP.
`BmpSaveImageToMemory`	Saves image to memory buffer in BMP format.

See Also

NImages Pro Library

6.9.1.1. BmpLoadImageFromFile Function

Loads image from BMP file.

```
NResult N_API BmpLoadImageFromFile(
        const NChar * szFileName,
        HNImage * pHImage
);
```

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Bmp Module | HNImage | BmpLoadImageFromMemory | BmpLoadImageFromHBitmap | BmpSaveImageToFile

6.9.1.2. BmpLoadImageFromHBitmap Function

Note
This function is available only on Windows.

Loads image from Windows HBITMAP.

```
NResult N_API BmpLoadImageFromHBitmap(
        NHandle handle,
        HNImage * pHImage
);
```

Parameters

handle	[in] Handle that specifies Windows HBITMAP.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*handle* or *pHImage* is NULL.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Bmp Module | HNImage | `BmpLoadImageFromFile` | `BmpLoadImageFromMemory` | `BmpSaveImageToHBitmap`

6.9.1.3. BmpLoadImageFromMemory Function

Loads image from memory buffer containing BMP file.

```
NResult N_API BmpLoadImageFromMemory(
        const void * buffer,
        NSizeType bufferLength,
        HNImage * pHImage
);
```

Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Bmp Module | HNImage | `BmpLoadImageFromFile` | `BmpLoadImageFromHBitmap` | `BmpSaveImageToMemory`

6.9.1.4. BmpSaveImageToFile Function

Saves image to file in BMP format.

```
NResult N_API BmpSaveImageToFile(
        HNImage hImage,
        const NChar * szFileName
);
```

Parameters

hImage	[in] Handle to image.
szFileName	[in] Points to string that specifies file name.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *szFileName* is NULL.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Bmp Module | HNImage | `BmpSaveImageToMemory` | `BmpSaveImageToHBitmap` | `BmpLoadImageFromFile`

6.9.1.5. BmpSaveImageToHBitmap Function

Note

This function is available only on Windows.

Saves image to Windows HBITMAP.

```
NResult N_API BmpSaveImageToHBitmap(
        HNImage hImage,
        NHandle * pHandle
);
```

Parameters

hImage	[in] Handle to image.
pHandle	[out] Pointer to NHandle that receives handle to created Windows HBITMAP.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pHandle* is NULL.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Bmp Module | HNImage | `BmpSaveImageToFile` | `BmpSaveImageToMemory` `BmpLoadImageFromHBitmap`

6.9.1.6. BmpSaveImageToMemory Function

Saves image to memory buffer in BMP format.

```
NResult N_API BmpSaveImageToMemory(
        HNImage hImage,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImage	[in] Handle to image.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage*, *pBuffer* or *pBufferLength* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory to allocate memory buffer.

Remarks

This is a low-level function and can be changed in future version of the library.

Memory buffer allocated by the function must be deallocated using `NFree` function when it is no longer needed.

See Also

Bmp Module | HNImage | `BmpSaveImageToFile` | `BmpSaveImageToHBitmap` | `BmpLoadImageFromMemory`

6.9.2. IHead Module

Provides functionality for loading and saving images in NIST IHead format.

Header file: `IHead.h`.

Functions

`IHeadLoadImageFromFile`	Loads image from NIST IHead file.
`IHeadLoadImageFromMemory`	Loads image from memory buffer containing NIST IHead file.

IHeadSaveImageToFile	Saves image to the file in NIST IHead format.
IHeadSaveImageToMemory	Saves image to the memory buffer in NIST IHead format.

See Also

NImages Pro Library

6.9.2.1. IHeadLoadImageFromFile Function

Loads image from NIST IHead file.

```
NResult N_API IHeadLoadImageFromFile(
        const NChar * szFileName,
        HNImage * pHImage
);
```

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

IHead Module | HNImage | IHeadLoadImageFromMemory

6.9.2.2. IHeadLoadImageFromMemory Function

Loads image from memory buffer containing NIST IHead file.

```
NResult N_API IHeadLoadImageFromMemory(
        const void * buffer,
        NSizeType bufferLength,
        HNImage * pHImage
);
```

Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

IHead Module | HNImage | IHeadLoadImageFromFile

6.9.2.3. IHeadSaveImageToFile Function

Saves image to the file in NIST IHead format.

```
NResult N_API IHeadSaveImageToFile(
        HNImage hImage,
        const NChar * szFileName
);
```

Parameters

hImage	[in] Handle to image.
szFileName	[in] Points to string that specifies file name.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *szFileName* is NULL.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

IHead Module | HNImage | IHeadSaveImageToMemory | IHeadLoadImageFrom-File

6.9.2.4. IHeadSaveImageToMemory Function

Saves image to the memory buffer in NIST IHead format.

```
NResult N_API IHeadSaveImageToMemory(
        HNImage hImage,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImage	[in] Handle to image.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage*, *pBuffer* or *pBufferLength* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory to allocate memory buffer.

Remarks

This is a low-level function and can be changed in future version of the library.

Memory buffer allocated by the function must be deallocated using `NFree` function when it is no longer needed.

See Also

IHead Module | HNImage | `IHeadSaveImageToFile` | `IHeadLoadImageFrom-Memory`

6.9.3. NGrayscaleImage Module

Provides functionality for managing 8-bit grayscale images.

Header file: `NGrayscaleImage.h`.

Functions

`NGrayscaleImageGetPixel`	Retrieves value of pixel at the specified co-ordinates in 8-bit grayscale image.

Copyright © 2006 Neurotechnologija

NGrayscaleImageSetPixel	Sets value of pixel at the specified coordinates in 8-bit grayscale image.

Remarks

This module provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to npfGrayscale.

See Also

NImages Pro Library | NImage Pro Module

6.9.3.1. NGrayscaleImageGetPixel Function

Retrieves value of pixel at the specified coordinates in 8-bit grayscale image.

```
NResult N_API NGrayscaleImageGetPixel(
        HNImage hImage,
        NUInt x,
        NUInt y,
        NByte * pValue
);
```

Parameters

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
pValue	[out] Points to NByte that receives pixel value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*x* is greater than or equal to image width.

Error Code	Condition
	- or - *y* is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to `npf-Grayscale`.

See Also

NGrayscaleImage Module | HNImage | `NGrayscaleImageSetPixel`

6.9.3.2. NGrayscaleImageSetPixel Function

Sets value of pixel at the specified coordinates in 8-bit grayscale image.

```
NResult N_API NGrayscaleImageSetPixel(
        HNImage hImage,
        NUInt x,
        NUInt y,
        NByte value
);
```

Parameters

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
value	[in] Specifies new pixel value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*x* is greater than or equal to image width.

Error Code	Condition
	- or - *y* is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to `npf-Grayscale`.

See Also

NGrayscaleImage Module | HNImage | `NGrayscaleImageGetPixel`

6.9.4. NImageFile Module

Provides functionality for reading image files in format-neutral way.

Header file: `NImageFile.h`.

Functions

`NImageFileClose`	Closes the file asociated with the image file.
`NImageFileCreate`	Opens image file of specified format.
`NImageFileFree`	Closes the image file. After the image file is closed the specified handle is no longer valid.
`NImageFileGetFormat`	Retrieves image format of the image file.
`NImageFileIsOpened`	Retrieves a value indicating whether the file associated with the image file is opened.
`NImageFileReadImage`	Reads image from the image file.

Types

HNImageFile	Handle to opened read-only image file.

See Also

NImages Pro Library | NImageFormat Pro Module | NImage Pro Module

Copyright © 2006 Neurotechnologija

6.9.4.1. NImageFileClose Function

Closes the file asociated with the image file.

```
NResult N_API NImageFileClose(
        HNImageFile hImageFile
);
```

Parameters

hImageFile	[in] Handle to image file.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFile* is NULL.

Remarks

If the file associated with the image file is already closed does nothing.

See Also

NImageFile Module | HNImageFile | `NImageFileIsOpened` | `NImageFileFree`

6.9.4.2. NImageFileCreate Function

Opens image file of specified format.

```
NResult N_API NImageFileCreate(
        const NChar * szFileName,
        HNImageFormat hImageFormat,
        HNImageFile * pHImageFile
);
```

Parameters

szFileName	[in] Points to string that specifies file name.
hImageFormat	[in] Handle to the image format of the file.

	Can be NULL.
pHImageFile	[out] Pointer to HNImageFile that receives handle to opened image file.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szFileName* or *pHImageFile* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid for specified image format.
N_E_NOT_SUPPORTED	*hImageFormat* is NULL and none of supported image formats is registered with file extension of *szFileName*. - or - *hImageFormat* is NULL and image format registered with file extension of *szFileName* does not support reading. - or - Image format specified by *hImageFormat* does not support reading.

Remarks

If *hImageFormat* is NULL image format is selected by file extension of *szFileName*.

Opened image file must be closed using NImageFileFree function.

This function does not check format of the file for all image formats. However format of the file is always checked in NImageFileReadImage function.

See Also

NImageFile Module | HNImageFile | NImageFileFree | NImageFileReadImage

6.9.4.3. NImageFileFree Function

Closes the image file. After the image file is closed the specified handle is no longer valid.

```
void N_API NImageFileFree(
        HNImageFile hImageFile
);
```

Parameters

hImageFile	[in] Handle to image file.

Remarks

If *hImageFile* is NULL does nothing.

See Also

NImageFile Module | HNImageFile | NImageFileCreate | NImageFileClose

6.9.4.4. NImageFileGetFormat Function

Retrieves image format of the image file.

```
NResult N_API NImageFileGetFormat(
        HNImageFile hImageFile,
        HNImageFormat * pValue
);
```

Parameters

hImageFile	[in] Handle to image file.
pValue	[out] Pointer to NImageFormat that receives image format of the image file.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFile* or *pValue* is NULL.

See Also

NImageFile Module | HNImageFile

6.9.4.5. NImageFileIsOpened Function

Retrieves a value indicating whether the file associated with the image file is opened.

```
NResult N_API NImageFileIsOpened(
        HNImageFile hImageFile,
        NBool * pValue
);
```

Parameters

hImageFile	[in] Handle to image file.
pValue	[out] Pointer to NBool that receives value indicating whether the file associated with the image file is opened.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFile* or *pValue* is NULL.

See Also

NImageFile Module | HNImageFile | `NImageFileClose`

6.9.4.6. NImageFileReadImage Function

Reads image from the image file.

```
NResult N_API NImageFileReadImage(
        HNImageFile hImageFile,
        HNImage * pHImage
);
```

Parameters

hImageFile	[in] Handle to image file.
pHImage	[out] Pointer to HNImage that receives handle to image read from the image file.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFile* or *pHImage* is NULL.
N_E_FORMAT	Format of file associated with the image file is invalid for the image format of the image file.
N_E_INVALID_OPERATION	File associated with the image file is closed.

Remarks

If all images are already read from the image file then handle to image returned via *pHImage* is NULL.

See Also

NImageFile Module | HNImageFile | HNImage | NImageFileIsOpened

6.9.5. NImageFormat Pro Module

Provides functionality for loading and saving images in format-neutral way.

Header file: NImageFormatPro.h (includes NImageFormat.h).

Functions

NImageFormatCanRead	Retrieves a value indicating whether the image format supports reading.
NImageFormatCanWrite	Retrieves a value indicating whether the image format supports writing.
NImageFormatCanWriteMultiple	Retrieves a value indicating whether the image format supports writing of multiple images.

Copyright © 2006 Neurotechnologija

NImageFormatGetBmp	Retrieves BMP image format.
NImageFormatGetDefaultFileExtension	Retrieves default file extension of the image format.
NImageFormatGetFileFilter	Retrieves file filter of the image format.
NImageFormatGetFormat	Retrieves supported image format with specified index.
NImageFormatGetFormatCount	Retrieves number of supported image formats.
NImageFormatGetIHead	Retrieves NIST IHead image format.
NImageFormatGetName	Retrieves name of the image format.
NImageFormatGetTiff	Retrieves TIFF image format.
NImageFormatGetWsq	Retrieves WSQ image format.
NImageFormatLoadImageFromFile	Loads image from file of specified image format.
NImageFormatLoadImageFromMemory	Loads image from the memory buffer containing file of specified image format.
NImageFormatOpenFile	Opens image file of specified format.
NImageFormatOpenFileFromMemory	Opens image file from the memory buffer containing file of specified format.
NImageFormatSaveImageToFile	Saves image to the file in specified format.
NImageFormatSaveImageToMemory	Saves image to the memory buffer in specified format.
NImageFormatSaveImagesToFile	Saves array of images to the file in specified format.
NImageFormatSaveImagesToMemory	Saves array of images to the memory buffer in specified format.
NImageFormatSelect	Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.

Types

HNImageFormat	Handle to image format.

See Also

NImages Pro Library | NImage Pro Module | NImageFile Module

6.9.5.1. NImageFormatCanRead Function

Retrieves a value indicating whether the image format supports reading.

```
NResult N_API NImageFormatCanRead(
        HNImageFormat hImageFormat,
        NBool * pValue
);
```

Parameters

hImageFormat	[in] Handle to image format.
pValue	[out] Pointer to NBool that receives value indicating whether the image format supports reading.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat* or *pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | `NImageFormatCanWrite`

6.9.5.2. NImageFormatCanWrite Function

Retrieves a value indicating whether the image format supports writing.

```
NResult N_API NImageFormatCanWrite(
        HNImageFormat hImageFormat,
        NBool * pValue
);
```

Parameters

hImageFormat	[in] Handle to image format.
pValue	[out] Pointer to NBool that receives value indicating whether the image format supports writing.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat* or *pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | `NImageFormatCanRead` | `NImageFormatCanWriteMultiple`

6.9.5.3. NImageFormatCanWriteMultiple Function

Retrieves a value indicating whether the image format supports writing of multiple images.

```
NResult N_API NImageFormatCanWriteMultiple(
        HNImageFormat hImageFormat,
        NBool * pValue
);
```

Parameters

hImageFormat	[in] Handle to image format.
pValue	[out] Pointer to NBool that receives value indicating whether the image format supports writing of multiple images.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat* or *pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | NImageFormatCanWrite

6.9.5.4. NImageFormatGetBmp Function

Retrieves BMP image format.

```
NResult N_API NImageFormatGetBmp(
        HNImageFormat * pValue
);
```

Parameters

pValue	[out] Pointer to HNImageFormat that receives handle to image format.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatGetIHead | NImageFormatGetTiff | NImageFormatGetWsq

6.9.5.5. NImageFormatGetDefaultFileExtension Function

Retrieves default file extension of the image format.

```
NResult N_API NImageFormatGetDefaultFileExtension(
        HNImageFormat hImageFormat,
        NChar * pValue
);
```

Parameters

hImageFormat	[in] Handle to image format.
pValue	[out] Pointer to string that receives default file extension of the image format. Can be NULL.

Return Values

If the function succeeds and *pValue* is NULL, the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat

6.9.5.6. NImageFormatGetFileFilter Function

Retrieves file filter of the image format.

```
NResult N_API NImageFormatGetFileFilter(
        HNImageFormat hImageFormat,
        NChar * pValue
);
```

Parameters

hImageFormat	[in] Handle to image format.
pValue	[out] Pointer to string that receives file filter of the image format. Can be NULL.

Return Values

If the function succeeds and *pValue* is NULL, the return value is length of the string (not

including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat

6.9.5.7. NImageFormatGetFormat Function

Retrieves supported image format with specified index.

```
NResult N_API NImageFormatGetFormat(
        NInt index,
        HNImageFormat * pValue
);
```

Parameters

index	[in] Specifies zero-based supported image format index to retrieve.
pValue	[out] Pointer to NImageFormat that receives image format.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*index* is less than zero or greater than or equal to supported image format count. See NImageFormatGetFormatCount.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatGetFormatCount

6.9.5.8. NImageFormatGetFormatCount Function

Retrieves number of supported image formats.

```
NResult N_API NImageFormatGetFormatCount(
        NInt * pValue
);
```

Parameters

pValue	[out] Pointer to NInt that receives number of supported image formats.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatGetFormat

6.9.5.9. NImageFormatGetIHead Function

Retrieves NIST IHead image format.

```
NResult N_API NImageFormatGetIHead(
        HNImageFormat * pValue
);
```

Parameters

pValue	[out] Pointer to HNImageFormat that receives handle to image format.

Return Values

Copyright © 2006 Neurotechnologija

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatGetBmp | NImageFormatGetTiff | NImageFormatGetWsq

6.9.5.10. NImageFormatGetName Function

Retrieves name of the image format.

```
NResult N_API NImageFormatGetName(
        HNImageFormat hImageFormat,
        NChar * pValue
);
```

Parameters

hImageFormat	[in] Handle to image format.
pValue	[out] Pointer to string that receives name of the image format. Can be NULL.

Return Values

If the function succeeds and *pValue* is NULL, the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat

Copyright © 2006 Neurotechnologija

6.9.5.11. NImageFormatGetTiff Function

Retrieves TIFF image format.

```
NResult N_API NImageFormatGetTiff(
        HNImageFormat * pValue
);
```

Parameters

pValue	[out] Pointer to HNImageFormat that receives handle to image format.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | `NImageFormatGetBmp` | `NImageFormatGetIHead` | `NImageFormatGetWsq`

6.9.5.12. NImageFormatGetWsq Function

Retrieves WSQ image format.

```
NResult N_API NImageFormatGetWsq(
        HNImageFormat * pValue
);
```

Parameters

pValue	[out] Pointer to HNImageFormat that receives handle to image format.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pValue* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatBmp | NImageFormatGetIHead | NImageFormatGetTiff

6.9.5.13. NImageFormatLoadImageFromFile Function

Loads image from file of specified image format.

```
NResult N_API NImageFormatLoadImageFromFile(
        HNImageFormat hImageFormat,
        const NChar * szFileName,
        HNImage * pHImage
);
```

Parameters

hImageFormat	[in] Handle to image format.
szFileName	[in] Points to string that specifies file name.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat*, *szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid for specified image format.

Error Code	Condition
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support reading.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | HNImage | NImageFormatLoadImageFromMemory | NImageFormatSaveImageToFile | NImageFormatOpenFile

6.9.5.14. NImageFormatLoadImageFromMemory Function

Loads image from the memory buffer containing file of specified image format.

```
NResult N_API NImageFormatLoadImageFromMemory(
        HNImageFormat hImageFormat,
        void * buffer,
        NSizeType bufferLength,
        HNImage * pHImage
);
```

Parameters

hImageFormat	[in] Handle to image format.
buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat* or *pHImage* is NULL. - or - *buffer* is NULL and *bufferLength* is

Error Code	Condition
	not equal to zero.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support reading.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | HNImage | NImageFormatLoadImageFromFile | NImageFormatSaveImageToMemory | NImageFormatOpenFileFromMemory

6.9.5.15. NImageFormatOpenFile Function

Opens image file of specified format.

```
NResult N_API NImageFormatOpenFile(
        HNImageFormat hImageFormat,
        const NChar * szFileName,
        HNImageFile * pHImageFile
);
```

Parameters

hImageFormat	[in] Handle to image format.
szFileName	[in] Points to string that specifies filename.
pHImageFile	[in] Pointer to HNImageFile that receives handle to opened image file.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat*, *szFileName* or *pHImageFile* is NULL.

Error Code	Condition
N_E_FORMAT	Format of file specified by *szFileName* is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by *hImage-Format* does not support reading.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | HNImageFile | NImageFormatOpenFileFromMemory | NImageFormatLoadImageFromFile

6.9.5.16. NImageFormatOpenFileFromMemory Function

Opens image file from the memory buffer containing file of specified format.

```
NResult N_API NImageFormatOpenFileFromMemory(
        HNImageFormat hImageFormat,
        const void * buffer,
        NSizeType bufferLength,
        HNImageFile * pHImageFile
);
```

Parameters

hImageFormat	[in] Handle to image format.
buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImageFile	[out] Pointer to HNImageFile that receives handle to opened image file.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat* or *pHImageFile* is NULL.

Error Code	Condition
	- or - *buffer* is NULL and *bufferLength* is not equal to zero.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid for specified image format.
N_E_NOT_SUPPORTED	Image format specified by *hImage-Format* does not support reading.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanRead | HNImageFile | NImageFormatOpenFile | NImageFormatLoadImageFromMemory

6.9.5.17. NImageFormatSaveImageToFile Function

Saves image to the file in specified format.

```
NResult N_API NImageFormatSaveImageToFile(
        HNImageFormat hImageFormat,
        HNImage hImage,
        const NChar * szFileName
);
```

Parameters

hImageFormat	[in] Handle to image format.
hImage	[in] Handle to image.
szFileName	[in] Points to string that specifies file name.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat*, *hImage* or *szFile-*

Error Code	Condition
	Name is NULL.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support writing.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanWrite | HNImage | NImageFormatSaveImageToMemory | NImageFormatLoadImageFromFile

6.9.5.18. NImageFormatSaveImageToMemory Function

Saves image to the memory buffer in specified format.

```
NResult N_API NImageFormatSaveImageToMemory(
        HNImageFormat hImageFormat,
        HNImage hImage,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImageFormat	[in] Handle to image format.
hImage	[in] Handle to image.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat*, *hImage*, *pBuffer* or *pBufferLength* is NULL.

Error Code	Condition
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support writing.

Remarks

Memory buffer allocated by the function must be deallocated using NFree function when it is no longer needed.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanWrite | HNImage | NImageFormatSaveImageToFile | NImageFormatLoadImageFromMemory

6.9.5.19. NImageFormatSaveImagesToFile Function

Saves array of images to the file in specified format.

```
NResult N_API NImageFormatSaveImagesToFile(
        HNImageFormat hImageFormat,
        NInt imageCount,
        HNImage * arHImages,
        const NChar * szFileName
);
```

Parameters

hImageFormat	[in] Handle to image format.
imageCount	[in] Specifies image count in the array.
arHImages	[in] Points to array of handles to images.
szFileName	[in] Points to string that specifies file name.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat*, *arHImages* or *szFileName* is NULL.

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	*imageCount* is less than or equal to zero.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support writing. - or - *imageCount* is greater than one and image format specified by *hImageFormat* does not support writing of multiple files.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanWriteMultiple | HNImage | NImageFormatSaveImagesToMemory | NImageFormatSaveImageToFile | NImageFormatLoadImageFromFile | NImageFormatOpenFile

6.9.5.20. NImageFormatSaveImagesToMemory Function

Saves array of images to the memory buffer in specified format.

```
NResult N_API NImageFormatSaveImagesToMemory(
        HNImageFormat hImageFormat,
        NInt imageCount,
        HNImage * arHImages,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImageFormat	[in] Handle to image format.
imageCount	[in] Specifies image count in the array.
arHImages	[in] Points to array of handles to images.
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImageFormat*, *arHImages*, *pBuffer* or *pBufferLength* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*imageCount* is less than or equal to zero.
N_E_NOT_SUPPORTED	Image format specified by *hImageFormat* does not support writing. - or - *imageCount* is greater than one and image format specified by *hImageFormat* does not support writing of multiple files.

Remarks

Memory buffer allocated by the function must be deallocated using NFree function when it is no longer needed.

See Also

NImageFormat Pro Module | HNImageFormat | NImageFormatCanWriteMultiple | HNImage | NImageFormatSaveImagesToFile | NImageFormatSaveImageToMemory | NImageFormatLoadImageFromMemory | NImageFormatOpenFileFromMemory

6.9.5.21. NImageFormatSelect Function

Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.

```
NResult N_API NImageFormatSelect(
        const NChar * szFileName,
        NFileAccess fileAccess,
        HNImageFormat * pHImageFormat
);
```

Parameters

szFileName	[in] Points to string that file name.
fileAccess	[in] Specifies that image format should sup-

	port reading, writing or both.
pHImageFormat	[out] Pointer to HNImageFormat that receives handle to image format.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*fileAccess* value is invalid.
N_E_ARGUMENT_NULL	*szFileName* or *pHImageFormat* is NULL.

Remarks

If none of supported image formats that supports reading/writing as specified by *fileAccess* is registered with file extension of *szFileName* then handle returned via *pHImageFormat* is NULL.

See Also

NImageFormat Pro Module | HNImageFormat | NFileAccess | NImageFormatGetFormatCount | NImageFormatGetFormat

6.9.6. NImage Pro Module

Provides functionality for managing images.

Header file: NImagePro.h (includes NImage.h).

Functions

NImageClone	Creates a new image that is a copy of specified image.
NImageCreate	Creates an image with specified pixel format, size, stride and resolution.
NImageCreateFromData	Creates an image with specified pixel format, size, stride and resolution and copies specified pixels to it.

NImageCreateFromFile	Creates (loads) an image from file of specified format.
NImageCreateFromImage	Creates an image from specified image with specified pixel format and stride.
NImageCreateFromImageEx	Creates an image from specified image with specified pixel format, stride and resolution.
NImageCreateWrapper	Creates an image wrapper for specified pixels with specified pixel format, size, stride and resolution.
NImageFree	Deletes the image. After the image is deleted the specified handle is no longer valid.
NImageGetHeight	Retrieves height of the image.
NImageGetHorzResolution	Retrieves horizontal resolution of the image.
NImageGetPixelFormat	Retrieves pixel format of the image.
NImageGetPixels	Retrieves pointer to memory block containing pixels of the image.
NImageGetSize	Retrieves size of memory block containing pixels of the image.
NImageGetStride	Retrieves stride (size of one row) of the image.
NImageGetVertResolution	Retrieves vertical resolution of the image.
NImageGetWidth	Retrieves width of the image.
NImageSaveToFile	Saves the image to the file of specified format.

Types

HNImage	Handle to image.

See Also

NImages Pro Library | NMonochromeImage Module | NGrayscaleImage Module | NRgbImage Module | NImageFormat Pro Module | NImageFile Module

6.9.6.1. NImageClone Function

Creates a new image that is a copy of specified image.

```
NResult N_API NImageClone(
        HNImage hImage,
        HNImage * pHClonedImage
);
```

Parameters

hImage	[in] Handle to the image.
pHClonedImage	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pHClonedImage* is NULL.

Remarks

Created image must be deleted using `NImageFree` function.

See Also

NImage Pro Module | HNImage | `NImageFree` | `NImageCreate`

6.9.6.2. NImageCreate Function

Creates an image with specified pixel format, size, stride and resolution.

```
NResult N_API NImageCreate(
        NPixelFormat pixelFormat,
        NUInt width,
        NUInt height,
        NSizeType stride,
        NFloat horzResolution,
        NFloat vertResolution,
        HNImage * pHImage
);
```

Parameters

pixelFormat	[in] Specifies pixel format of the image.
width	[in] Specifies width of the image.
height	[in] Specifies height of the image.
stride	[in] Specifies stride of the image. Can be zero.
horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
pHImage	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*pixelFormat* has invalid value. - or - *stride* is not zero and is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	*pHImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is less than zero.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride function.

Created image must be deleted using NImageFree function.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

NImage Pro Module | HNImage | NImageFree | NImageCreateWrapper | NImageCreateFromData | NImageCreateFromImage | NImageCreateFromFile | NImageClone | NImageGetStride

6.9.6.3. NImageCreateFromData Function

Creates an image with specified pixel format, size, stride and resolution and copies specified pixels to it.

```
NResult N_API NImageCreateFromData(
        NPixelFormat pixelFormat,
        NUInt width,
        NUInt height,
        NSizeType stride,
        NFloat horzResolution,
        NFloat vertResolution,
        NSizeType srcStride,
        const void * srcPixels,
        HNImage * pHImage
);
```

Parameters

pixelFormat	[in] Specifies pixel format of the image.
width	[in] Specifies width of the image.
height	[in] Specifies height of the image.
stride	[in] Specifies stride of the image. Can be zero.
horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
srcStride	[in] Specifies stride of pixels to be copied to the image.

Copyright © 2006 Neurotechnologija

srcPixels	[in] Points to memory block containing pixels that to be copied to the image.
pHImage	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*pixelFormat* has invalid value. - or - *stride* is not zero and is less than minimal value for specified pixel format and width. - or - *srcStride* is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	*srcPixels* or *pHImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is less than zero.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride function.

Format of memory block *srcPixels* points to must be the same as described in NImageGetPixels function, only stride is equal to *srcStride*.

Created image must be deleted using NImageFree function.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

Copyright © 2006 Neurotechnologija

See Also

NImage Pro Module | HNImage | `NImageFree` | `NImageCreate` | `NImageCreate-Wrapper` | `NImageGetStride` | `NImageGetPixels`

6.9.6.4. NImageCreateFromFile Function

Creates (loads) an image from file of specified format.

```
NResult N_API NImageCreateFromFile(
        const NChar * szFileName,
        HNImageFormat hImageFormat,
        HNImage * pHImage
);
```

Parameters

szFileName	[in] Points to string that specifies file name.
hImageFormat	[in] Handle to the image format of the file. Can be NULL.
pHImage	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid for specified image format.
N_E_NOT_SUPPORTED	*hImageFormat* is NULL and none of supported image formats is registered with file extension of *szFileName*. - or - *hImageFormat* is NULL and image format registered with file extension of *szFileName* does not support reading.

Error Code	Condition
	- or - Image format specified by *hImageFormat* does not support reading.

Remarks

If *hImageFormat* is NULL image format is selected by file extension of *szFileName*.

Created image must be deleted using NImageFree function.

See Also

NImage Pro Module | HNImage | NImageFree | NImageCreate | NImageFormatCanRead

6.9.6.5. NImageCreateFromImage Function

Creates an image from specified image with specified pixel format and stride.

```
NResult N_API NImageCreateFromImage(
        NPixelFormat pixelFormat,
        NSizeType stride,
        HNImage hSrcImage,
        HNImage * pHImage
);
```

Parameters

pixelFormat	[in] Specifies pixel format of the image.
stride	[in] Specifies stride of the image. Can be zero.
hSrcImage	[in] Handle to image used as source for the image.
pHImage	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Copyright © 2006 Neurotechnologija

Error Code	Condition
N_E_ARGUMENT	*pixelFormat* has invalid value. - or - *stride* is not zero and is less than minimal value for specified pixel format and source image width.
N_E_ARGUMENT_NULL	*hSrcImage* or *pHImage* is NULL.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride function.

Created image must be deleted using NImageFree function.

See Also

NImage Pro Module | HNImage | NImageFree | NImageCreate | NImageCreateFromImageEx | NImageClone | NImageGetStride

6.9.6.6. NImageCreateFromImageEx Function

Creates an image from specified image with specified pixel format, stride and resolution.

```
NResult N_API NImageCreateFromImageEx(
        NPixelFormat pixelFormat,
        NSizeType stride,
        NFloat horzResolution,
        NFloat vertResolution,
        HNImage hSrcImage,
        HNImage * pHImage
);
```

Parameters

pixelFormat	[in] Specifies pixel format of the image.
stride	[in] Specifies stride of the image. Can be zero.
horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.

hSrcImage	[in] Handle to image used as source for the image.
pHImage	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*pixelFormat* has invalid value. - or - *stride* is not zero and is less than minimal value for specified pixel format and source image width.
N_E_ARGUMENT_NULL	*hSrcImage* or *pHImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*horzResolution* or *vertResolution* is less than zero.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see NImageGetStride function.

Created image must be deleted using NImageFree function.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

NImage Pro Module | HNImage | NImageFree | NImageCreate | NImageCreateFromImage | NImageClone | NImageGetStride

6.9.6.7. NImageCreateWrapper Function

Creates an image wrapper for specified pixels with specified pixel format, size, stride and resolution.

```
NResult N_API NImageCreateWrapper(
        NPixelFormat pixelFormat,
        NUInt width,
        NUInt height,
        NSizeType stride,
        NFloat horzResolution,
        NFloat vertResolution,
        void * pixels,
        NBool ownsPixels,
        HNImage * pHImage
);
```

Parameters

pixelFormat	[in] Specifies pixel format of the image.
width	[in] Specifies width of the image.
height	[in] Specifies height of the image.
stride	[in] Specifies stride of the image.
horzResolution	[in] Specifies horizontal resolution in pixels per inch of the image.
vertResolution	[in] Specifies vertical resolution in pixels per inch of the image.
pixels	[in] Points to memory block containing pixels for the image.
ownsPixels	[in] Specifies whether pixels will be automatically deleted with the image (if set to NTrue).
pHImage	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*pixelFormat* has invalid value. - or -

Error Code	Condition
	stride is less than minimal value for specified pixel format and width.
N_E_ARGUMENT_NULL	*pixels* or *pHImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is less than zero.

Remarks

For more information on image stride see NImageGetStride function.

Format of memory block *pixels* points to must be the same as described in NImageGetPixels function.

Created image must be deleted using NImageFree function.

pixels must not be deleted during lifetime of the image. If *ownsPixels* is NTrue then *pixels* will be automatically deleted with the image.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

NImage Pro Module | HNImage | NImageFree | NImageCreate | NImageCreateFromData | NImageGetStride | NImageGetPixels

6.9.6.8. NImageFree Function

Deletes the image. After the image is deleted the specified handle is no longer valid.

```
void N_API NImageFree(
        HNImage hImage
);
```

Parameters

hImage	[in] Handle to the image.

Remarks

If *hImage* is NULL does nothing.

See Also

NImage Pro Module | HNImage | NImageCreate

6.9.6.9. NImageGetHeight Function

Retrieves height of the image.

```
NResult N_API NImageGetHeight(
        HNImage hImage,
        NUInt * pValue
);
```

Parameters

hImage	[in] Handle to the image.
pValue	[out] Pointer to NUInt that receives height of the image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

See Also

NImage Pro Module | HNImage | NImageGetWidth

6.9.6.10. NImageGetHorzResolution Function

Retrieves horizontal resolution of the image.

```
NResult N_API NImageGetHorzResolution(
        HNImage hImage,
        NFloat * pValue
);
```

Copyright © 2006 Neurotechnologija

Parameters

hImage	[in] Handle to the image.
pValue	[out] Pointer to NFloat that receives horizontal resolution in pixels per inch of the image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

Remarks

Horizontal resolution equal to zero means that it is not applicable for the image.

See Also

NImage Pro Module | HNImage | `NImageGetVertResolution`

6.9.6.11. NImageGetPixelFormat Function

Retrieves pixel format of the image.

```
NResult N_API NImageGetPixelFormat(
        HNImage hImage,
        NPixelFormat * pValue
);
```

Parameters

hImage	[in] Handle to the image.
pValue	[out] Pointer to NPixelFormat that receives pixel format of the image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

See Also

NImage Pro Module | HNImage | NPixelFormat

6.9.6.12. NImageGetPixels Function

Retrieves pointer to memory block containing pixels of the image.

```
NResult N_API NImageGetPixels(
        HNImage hImage,
        void * * pValue
);
```

Parameters

hImage	[in] Handle to the image.
pValue	[out] Pointer to void * that receives pointer to memory block containing pixels of the image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

Remarks

Memory block containing image pixels is organized as image height rows following each other in top-to-bottom order. Each row occupies image stride bytes and is organized as image width pixels following each other in right-to-left order. Each pixel is described by image

pixel format.

For more information see NImageGetPixelFormat, NImageGetWidth, NImage-GetHeight, NImageGetStride, and NImageGetSize functions.

See Also

NImage Pro Module | HNImage | NImageGetPixelFormat | NImageGetWidth | NImageGetHeight | NImageGetStride | NImageGetSize

6.9.6.13. NImageGetSize Function

Retrieves size of memory block containing pixels of the image.

```
NResult N_API NImageGetSize(
        HNImage hImage,
        NSizeType * pValue
);
```

Parameters

hImage	[in] Handle to the image.
pValue	[out] Pointer to NSizeType that receives size of memory block containing pixels of the image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

Remarks

Size of memory block containing image pixels is equal to image height multiplied by image stride. For more information see NImageGetHeight and NImageGetStride functions.

See Also

NImage Pro Module | HNImage | NImageGetHeight | NImageGetStride

6.9.6.14. NImageGetStride Function

Retrieves stride (size of one row) of the image.

```
NResult N_API NImageGetStride(
        HNImage hImage,
        NSizeType * pValue
);
```

Parameters

hImage	[in] Handle to the image.
pValue	[out] Pointer to NSizeType that receives stride of the image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

Remarks

Stride (size of one row) of the image depends on image pixel format and width. It can not be less than value obtained with NPixelFormatGetRowSize macro with arguments obtained with `NImageGetPixelFormat` and `NImageGetWidth` functions.

See Also

NImage Pro Module | HNImage | NPixelFormatGetRowSize | `NImageGetPixelFormat` | `NImageGetWidth` | `NImageGetSize`

6.9.6.15. NImageGetVertResolution Function

Retrieves vertical resolution of the image.

```
NResult N_API NImageGetVertResolution(
        HNImage hImage,
        NFloat * pValue
);
```

Copyright © 2006 Neurotechnologija

Parameters

hImage	[in] Handle to the image.
pValue	[out] Pointer to NFloat that receives vertical resolution in pixels per inch of the image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

Remarks

Vertical resolution equal to zero means that it is not applicable for the image.

See Also

NImage Pro Module | HNImage | NImageGetHorzResolution

6.9.6.16. NImageGetWidth Function

Retrieves width of the image.

```
NResult N_API NImageGetWidth(
        HNImage hImage,
        NUInt * pValue
);
```

Parameters

hImage	[in] Handle to the image.
pValue	[out] Pointer to NUInt that receives width of the image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

See Also

NImage Pro Module | HNImage | `NImageGetHeight` | `NImageGetStride`

6.9.6.17. NImageSaveToFile Function

Saves the image to the file of specified format.

```
NResult N_API NImageSaveToFile(
        HNImage hImage,
        const NChar * szFileName,
        HNImageFormat hImageFormat
);
```

Parameters

hImage	[in] Handle to NImage object.
szFileName	[in] Points to string that specifies file name.
hImageFormat	[in] Handle to the image format of the file. Can be NULL.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *szFileName* is NULL.
N_E_NOT_SUPPORTED	*hImageFormat* is NULL and none of supported image formats is registered with file extension of *szFileName*. - or - *hImageFormat* is NULL and image format registered with file extension of *sz-*

Error Code	Condition
	FileName does not support writing. - or - Image format specified by *hImage-Format* does not support writing.

Remarks

If *hImageFormat* is NULL image format is selected by file extension of *szFileName*.

See Also

NImage Pro Module | HNImage | NImageCreateFromFile | NImageFormatCan-Write

6.9.7. NImages Pro Module

Provides library registration and other additional functionality.

Header file: NImagesPro.h (includes NImages.h).

Functions

NImagesGetGrayscaleColorWrap-per	Creates color wrapper for grayscale image.
NImagesIsRegistered	Checks if NImages Pro library is registered.
NImagesRegister	Registers NImages Pro library.
NImagesUnregister	Unregisters NImages Pro library.

See Also

NImages Pro Library

6.9.7.1. NImagesGetGrayscaleColorWrapper Function

Creates color wrapper for grayscale image.

```
NResult N_API NImagesGetGrayscaleColorWrapper(
        HNImage hImage,
        NRgb minColor,
        NRgb maxColor,
```

```
        HNImage * pHDstImage
);
```

Parameters

hImage	[in] Handle to image.
minColor	[in] Specifies color to be used for black color.
maxColor	[in] Specifies color to be used for white color.
pHDstImage	[out] Pointer to HNImage that receives handle to created image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Image specified by *hImage* has non-grayscale pixel format (not npfGrayscale or npfMonochrome).
N_E_ARGUMENT_NULL	*hImage* or *pHDstImage* is NULL.

Remarks

Created image must be deleted using NImageFree function.

Created image is a thin wrapper for specified grayscale image. Therefore *hImage* must not be freed before created image.

Gray values in source image are replaced with according RGB values from range [*minColor*, *maxColor*] in created image.

See Also

NImages Pro Module | HNImage | NImageFree

6.9.7.2. NImagesIsRegistered Function

Checks if NImages Pro library is registered.

```
NBool N_API NImagesIsRegistered(void);
```

Return Values

NTrue if library is registered, NFalse otherwise.

See Also

NImages Pro Module | `NImagesRegister` | `NImagesUnregister`

6.9.7.3. NImagesRegister Function

Registers NImages Pro library.

```
NResult N_API NImagesRegister(
        const NChar * szServer
);
```

Parameters

szServer	[in] Pointer to string that specifies license manager server name or IP address (if registering with LAN License Manager), any string (if registering with HASP) or serial number and registration key separated by '@' (if registering using PC or U.are.U protection).

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Registering with PC or U.are.U protection and serial number or registration key has invalid value.
N_E_ARGUMENT_NULL	*szServer* is NULL.
N_E_LM_CONNECTION	There was error during connection or communication with license manager server.
N_E_LM_NO_MORE_LICENSE	Registered license count achieved license manager license limit for NImages Pro lib-

Error Code	Condition
	rary.

Remarks

If registering with HASP it is recommended that *szServer* should point to empty string.

If registering with PC/U.are.U protection *szServer* should point to string in format "*serial_number@registration_key*".

When library is no longer needed (possibly during application shutdown) it must be unregistered using NImagesUnregister function.

See Also

NImages Pro Module | NImagesUnregister | NImagesIsRegister

6.9.7.4. NImagesUnregister Function

Unregisters NImages Pro library.

```
void N_API NImagesUnregister(void);
```

See Also

NImages Pro Module | NImagesRegister | NImagesIsRegistered

6.9.8. NMonochromeImage Module

Provides functionality for managing 1-bit monochrome images.

Header file: NMonochromeImage.h.

Functions

NMonochromeImageGetPixel	Retrieves value of pixel at the specified coordinates in 1-bit monochrome image.
NMonochromeImageSetPixel	Sets value of pixel at the specified coordinates in 1-bit monochrome image.

Remarks

This module provides advanced functionality, such as individual pixel value retrieval for im-

age with pixel format equal to `npfMonochrome`.

See Also

NImages Pro Library | NImage Pro Module

6.9.8.1. NMonochromeImageGetPixel Function

Retrieves value of pixel at the specified coordinates in 1-bit monochrome image.

```
NResult N_API NMonochromeImageGetPixel(
        HNImage hImage,
        NUInt x,
        NUInt y,
        NBool * pValue
);
```

Parameters

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
pValue	[out] Points to NBool that receives pixel value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*x* is greater than or equal to image width. - or - *y* is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to `npfMonochrome`.

Copyright © 2006 Neurotechnologija

Remarks

If pixel is black then value *pValue* points to receives NFalse and if it is white then value receives NTrue.

See Also

NMonochromeImage Module | HNImage | `NMonochromeImageSetPixel`

6.9.8.2. NMonochromeImageSetPixel Function

Sets value of pixel at the specified coordinates in 1-bit monochrome image.

```
NResult N_API NMonochromeImageSetPixel(
        HNImage hImage,
        NUInt x,
        NUInt y,
        NBool value
);
```

Parameters

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
value	[in] Specifies new pixel value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*x* is greater than or equal to image width. - or - *y* is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to `npfMonochrome`.

Remarks

If *value* is NFalse then pixel will be black and if it is NTrue then pixel will be white.

See Also

NMonochromeImage Module | HNImage | NMonochromeImageGetPixel

6.9.9. NPixelFormat Module

Provides functionality for work with image pixel format.

Header file: NPixelFormat.h.

Functions

NPixelFormatGetBitsPerPixel-Func	Used internally in NPixelFormatGetBitsPer-Pixel macro.
NPixelFormatIsValid	Checks if specified pixel format is valid.

Structures

NRgb	Represents an RGB color.

Enumerations

NPixelFormat	Specifies pixel format of each pixel in the image.

Macros

NCalcRowSize	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
NCalcRowSizeEx	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel and alignment.
NPixelFormatGetBitsPerPixel	Retrieves number of bits used to store a pixel from NPixelFormat.
NPixelFormatGetRowSize	Calculates number of bytes needed to store

Copyright © 2006 Neurotechnologija

	line of specified length of pixels with specified NPixelFormat.
NPixelFormatGetRowSizeEx	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat and alignment.
NRgbConst	Makes NRgb constant with field values provided.

See Also

NImages Pro Library

6.9.9.1. NPixelFormat Enumeration

Specifies pixel format of each pixel in the image.

```
typedef enum NPixelFormat_ { } NPixelFormat;
```

Members

`npfGrayscale`	Each pixel value is stored in 8 bits representing 256 shades of gray.
`npfMonochrome`	Each pixel value is stored in 1 bit representing either black or white color.
`npfRgb`	Each pixel value is stored in 24 bits consisting of three 8-bit values representing red, green and blue color components.

Remarks

Image pixel format is not limited to members of this enumeration. However only these members are provided for usage with this product.

See Also

NPixelFormat Module | HNImage

6.9.9.2. NRgb Structure

Represents an RGB color.

```
typedef struct NRgb_ { } NRgb;
```

Fields

Blue	Blue component value of this NRgb.
Green	Green component value of this NRgb.
Red	Red component value of this NRgb.

See Also

NPixelFormat Module

6.9.9.2.1. NRgb.Blue Field

Blue component value of this NRgb.

```
NByte Blue;
```

See Also

NRgb Structure

6.9.9.2.2. NRgb.Green Field

Green component value of this NRgb.

```
NByte Green;
```

See Also

NRgb Structure

6.9.9.2.3. NRgb.Red Field

Red component value of this NRgb.

```
NByte Red;
```

See Also

NRgb Structure

6.9.10. NRgbImage Module

Provides functionality for managing 24-bit RGB images.

Header file: NRgbImage.h.

Functions

NRgbImageGetPixel	Retrieves value of pixel at the specified coordinates in 24-bit RGB image.
NRgbImageSetPixel	Sets value of pixel at the specified coordinates in 24-bit RGB image.

Remarks

This module provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to npfRgb.

See Also

NImages Pro Library | NImage Pro Module

6.9.10.1. NRgbImageGetPixel Function

Retrieves value of pixel at the specified coordinates in 24-bit RGB image.

```
NResult N_API NRgbImageGetPixel(
        HNImage hImage,
        NUInt x,
        NUInt y,
        NRgb * pValue
);
```

Parameters

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
pValue	[out] Pointer to NRgb that receives pixel value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*x* is greater than or equal to image width. - or - *y* is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to npfRgb.

See Also

NRgbImage Module | HNImage | NRgb | NRgbImageSetPixel

6.9.10.2. NRgbImageSetPixel Function

Sets value of pixel at the specified coordinates in 24-bit RGB image.

```
NResult N_API NRgbImageSetPixel(
        HNImage hImage,
        NUInt x,
        NUInt y,
        const NRgb * pValue
);
```

Parameters

hImage	[in] Handle to image.
x	[in] Specifies x-coordinate of the pixel.
y	[in] Specifies y-coordinate of the pixel.
pValue	[in] Pointer to NRgb that specifies new pixel value.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *pValue* is NULL.

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	*x* is greater than or equal to image width. - or - *y* is greater than or equal to image height.
N_E_FORMAT	Image pixel format is not equal to `npfRgb`.

See Also

NRgbImage Module | HNImage | NRgb | `NRgbImageGetPixel`

6.9.11. Tiff Module

Provides functionality for loading images in TIFF format.

Header file: `Tiff.h`.

Functions

`TiffLoadImageFromFile`	Loads image from TIFF file.
`TiffLoadImageFromMemory`	Loads image from memory buffer containing TIFF file.

See Also

NImages Pro Library

6.9.11.1. TiffLoadImageFromFile Function

Loads image from TIFF file.

```
NResult N_API TiffLoadImageFromFile(
        const NChar * szFileName,
        HNImage * pHImage
);
```

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Tiff Module | HNImage | TiffLoadImageFromMemory

6.9.11.2. TiffLoadImageFromMemory Function

Loads image from memory buffer containing TIFF file.

```
NResult N_API TiffLoadImageFromMemory(
        const void * buffer,
        NSizeType bufferLength,
        HNImage * pHImage
);
```

Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Tiff Module | HNImage | TiffLoadImageFromFile

6.9.12. Wsq Module

Provides functionality for loading and saving images in WSQ format.

Header file: Wsq.h.

Functions

WsqLoadImageFromFile	Loads image from WSQ file.
WsqLoadImageFromMemory	Loads image from memory buffer containing WSQ file.
WsqSaveImageToFile	Saves image to file in WSQ format with specified bitrate.
WsqSaveImageToMemory	Saves image to memory buffer in WSQ format with specified bitrate.

Macros

WSQ_DEFAULT_BIT_RATE	Specifies default bit rate (compression level).

See Also

NImages Pro Library

6.9.12.1. WsqLoadImageFromFile Function

Loads image from WSQ file.

```
NResult N_API WsqLoadImageFromFile(
        const NChar * szFileName,
        HNImage * pHImage
);
```

Parameters

szFileName	[in] Points to string that specifies file name.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szFileName* or *pHImage* is NULL.
N_E_FORMAT	Format of file specified by *szFileName* is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Wsq Module | HNImage | `WsqLoadImageFromMemory` | `WsqSaveImageToFile`

6.9.12.2. WsqLoadImageFromMemory Function

Loads image from memory buffer containing WSQ file.

```
NResult N_API WsqLoadImageFromMemory(
        const void * buffer,
        NSizeType bufferLength,
        HNImage * pHImage
```

```
);
```

Parameters

buffer	[in] Pointer to memory buffer.
bufferLength	[in] Length of memory buffer.
pHImage	[out] Pointer to HNImage that receives handle to loaded image.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL and *bufferLength* is not equal to zero. - or - *pHImage* is NULL.
N_E_FORMAT	Format of file contained in buffer specified by *buffer* is invalid.

Remarks

This is a low-level function and can be changed in future version of the library.

See Also

Wsq Module | HNImage | `WsqLoadImageFromFile` | `WsqSaveImageToMemory`

6.9.12.3. WsqSaveImageToFile Function

Saves image to file in WSQ format with specified bitrate.

```
NResult N_API WsqSaveImageToFile(
        HNImage hImage,
        NFloat bitRate,
        const NChar * szFileName
);
```

Parameters

hImage	[in] Handle to image.
bitRate	[in] Specifies bit rate (compression level).
szFileName	[in] Points to string that specifies file name.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage* or *szFileName* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*bitRate* is less than or equal to zero.

Remarks

This is a low-level function and can be changed in future version of the library.

bitRate can be set to WSQ_DEFAULT_BIT_RATE (0.75) that corresponds to 15:1 compression. The lower the bit rate is, the higher is the compression level and vice versa. Another common bit rate is 2.25 for 5:1 compression.

See Also

Wsq Module | HNImage | WSQ_DEFAULT_BIT_RATE | `WsqSaveImageToMemory` | `WsqLoadImageFromFile`

6.9.12.4. WsqSaveImageToMemory Function

Saves image to memory buffer in WSQ format with specified bitrate.

```
NResult N_API WsqSaveImageToMemory(
        HNImage hImage,
        NFloat bitRate,
        void * * pBuffer,
        NSizeType * pBufferLength
);
```

Parameters

hImage	[in] Handle to image.

bitRate	[in] Specifies bit rate (compression level).
pBuffer	[out] Pointer to void * that receives pointer to allocated memory buffer.
pBufferLength	[out] Pointer to NSizeType that receives size of allocated memory buffer.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hImage*, *pBuffer* or *pBufferLength* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*bitRate* is less than or equal to zero.
N_E_OUT_OF_MEMORY	There was not enough memory to allocate memory buffer.

Remarks

This is a low-level function and can be changed in future version of the library.

Memory buffer allocated by the function must be deallocated using `NFree` function when it is no longer needed.

See Also

Wsq Module | HNImage | WSQ_DEFAULT_BIT_RATE | `WsqSaveImageToFile` | `WsqLoadImageFromMemory`

6.10. NMatcher Library

Provides functionality for comparing Neurotechnologija Templates.

Import library (Windows): `NMatcher.dll.lib`.

DLL (Windows): `NMatcher.dll`.

Shared object (Linux): `libNMatcher.so`.

Requirements (Windows):

- `NCore.dll`.
- `NTemplate.dll`.
- `NfsMatcher.dll`.

Requirements (Linux):

- `libNCore.so`.
- `libNTemplate.so`.
- `libNfsMatcher.dll`.

Modules

NMatcher	Provides functionality for comparing Neurotechnologija Templates (NTemplates), encapsulated in Neurotechnologija Matcher (NMatcher) object.

6.10.1. NMatcher Module

Provides functionality for comparing Neurotechnologija Templates (NTemplates), encapsulated in Neurotechnologija Matcher (NMatcher) object.

Header file: `NMatcher.h` (includes `NMatcherParams.h` and `NMMatchDetails.h`).

Functions

`NMCopyParameters`	Copies parameter values from one NMatcher to another.
`NMCreate`	Creates a NMatcher.
`NMFree`	Deletes the NMatcher. After the object is deleted the specified handle is no longer valid.
`NMGetParameter`	Retrieves value of the specified parameter of the specified NMatcher.
`NMIdentifyEnd`	Ends identification using the specified NMatcher.
`NMIdentifyNext`	Compares the specified packed NTemplate with the one identification was started with using the specified NMatcher.
`NMIdentifyStart`	Starts identification with the specified packed NTemplate using the specified

Copyright © 2006 Neurotechnologija

	NMatcher.
NMIsRegistered	Checks if NMatcher library is registered.
NMMatchDetailsFree	Deletes NMMatchDetailsFree.
NMRegister	Registers NMatcher library.
NMReset	Sets default values for all parameters of the specified NMatcher.
NMSetParameter	Sets value of the specified parameter of the specified NMatcher.
NMUnregister	Unregisters NMatcher library.
NMVerify	Compares two packed NTemplate using the specified NMatcher.

Structures

NMMatchDetails	Represents details of matching performed with NMMatcher.

Types

HNMatcher	Handle to NMatcher object.

Macros

NM_PART_NONE	Identifier specifying that retrieving or setting parameter for NMatcher.
NM_PART_FINGERS	Identifier specifying that retrieving or setting parameter for fingers matcher (NfsMatcher) contained in NMatcher.
NMP_COPYRIGHT	Identifier specifying library copyright static read-only parameter of type N_TYPE_STRING.
NMP_MATCHING_THRESHOLD	Identifier specifying matching threshold (biggest allowed FAR) parameter of type N_TYPE_INT. Parameter value is equal to -12 * log10(FAR) and must be not less than

	zero (for example, 48 for FAR = 0.01%).
NMP_NAME	Identifier specifying library name static read-only parameter of type N_TYPE_STRING.
NMP_VERSION_HIGH	Identifier specifying high part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
NMP_VERSION_LOW	Identifier specifying low part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

See Also

NMatcher Library

6.10.1.1. NMCopyParameters Function

Copies parameter values from one NMatcher to another.

```
NResult N_API NMCopyParameters(
        HNMatcher hDstMatcher,
        HNMatcher hSrcMatcher
);
```

Parameters

hDstMatcher	[in] Handle to the destination NMatcher object.
hSrcMatcher	[in] Handle to the source NMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hDstMatcher* or *hSrcMatcher* is NULL.
N_E_INVALID_OPERATION	Identification is started on *hDstMatcher*.

See Also

NMatcher Module | HNMatcher

6.10.1.2. NMCreate Function

Creates a NMatcher.

```
NResult N_API NMCreate(
        HNMatcher * pHMatcher
);
```

Parameters

pHMatcher	[out] Pointer to HNMatcher that receives handle to created NMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pHMatcher* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NMFree` function.

See Also

NMatcher Module | HNMatcher | `NMFree`

6.10.1.3. NMFree Function

Deletes the NMatcher. After the object is deleted the specified handle is no longer valid.

```
void N_API NMFree(
        HNMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to NMatcher object.

Remarks

If *hMatcher* is NULL, does nothing.

See Also

NMatcher Module | HNMatcher

6.10.1.4. NMGetParameter Function

Retrieves value of the specified parameter of the specified NMatcher.

```
NResult N_API NMGetParameter(
        HNMatcher hMatcher,
        NUShort partId,
        NUInt parameterId,
        void * pValue
);
```

Parameters

hMatcher	[in] Handle to the NMatcher object. Can be NULL if retrieving static parameter value.
partId	[in] Identifier of the part of the matcher.
parameterId	[in] Identifier of the parameter to retrieve.
pValue	[out] Pointer to variable that receives parameter value.

Return Values

If the function succeeds and *parameterId* specifies a N_TYPE_STRING type parameter,

and *pValue* is NULL, the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*partId* is invalid.
N_E_ARGUMENT_NULL	*parameterId* specifies a non-static parameter and *hMatcher* is NULL. - or - *parameterId* specifies a non-N_TYPE_STRING type parameter and *pValue* is NULL.
N_E_PARAMETER	*parameterId* is invalid.

Remarks

partId can be one of the following:

- NM_PART_FINGERS + any *partId*`NfsmGetParameter` function supports
- NM_PART_NONE

If *partId* is NM_PART_FINGERS + any *partId* `NfsmGetParameter` function supports, *parameterId* must be any `NfsmGetParameter` function supports except for NFSMP_MATCHING_THRESHOLD.

If *partId* is NM_PART_NONE, the following values can be used for *parameterId*:

- NFSMP_COPYRIGHT
- NFSMP_MATCHING_THRESHOLD
- NFSMP_NAME
- NFSMP_VERSION_HIGH
- NFSMP_VERSION_LOW

To learn the type of the parameter pass value obtained with NParameterMakeId macro using N_PC_TYPE_ID code and the parameter id via *parameterId* parameter and pointer to NInt that will receive one of N_TYPE_XXX via *pValue* parameter. *hMatcher* can be NULL in this case.

See Also

NMatcher Module | HNMatcher | NfsmGetParameter | NMSetParameter

6.10.1.5. NMIdentifyEnd Function

Ends identification using the specified NMatcher.

```
NResult N_API NMIdentifyEnd(
        HNMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to the NMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* is NULL.
N_E_INVALID_OPERATION	Identification is not started.

See Also

NMatcher Module | HNMatcher | NMIdentifyStart | NMIdentifyNext

6.10.1.6. NMIdentifyNext Function

Compares the specified packed NTemplate with the one identification was started with using the specified NMatcher.

```
NResult N_API NMIdentifyNext(
        HNMatcher hMatcher,
        const void * templ,
        NSizeType templSize,
        NMMatchDetails * * ppMatchDetails,
        NInt * pScore
);
```

Parameters

hMatcher	[in] Handle to the NMatcher object.

Copyright © 2006 Neurotechnologija

templ	[in] Pointer to memory buffer containing packed NTemplate.
templSize	[in] Size of packed NTemplate.
pMatchDetails	[in, out] Pointer to NMMatchDetails that is filled with details of the matching. Can be NULL.
pScore	[out] Pointer to NInt that receives similarity score.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher*, *templ*, or *pScore* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NTemplate format.
N_E_INVALID_OPERATION	Identification is not started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see NMP_MATCHING_THRESHOLD and NMSetParameter function), and is greater than or equal to matching threshold otherwise.

If *pMatchDetails* is not NULL it should be a pointer obtained using NMIdentifyStart function.

See Also

NMatcher Module | HNMatcher | NMMatchDetails | NMP_MATCHING_THRESHOLD | NMSetParameter | NMIdentifyStart | NMIdentifyEnd

6.10.1.7. NMIdentifyStart Function

Starts identification with the specified packed NTemplate using the specified NMatcher.

```
NResult N_API NMIdentifyStart(
```

```
        HNMatcher hMatcher,
        const void * templ,
        NSizeType templSize,
        NMMatchDetails * * ppMatchDetails
);
```

Parameters

hMatcher	[in] Handle to the NMatcher object.
templ	[in] Pointer to memory buffer containing packed NTemplate.
templSize	[in] Size of packed NTemplate.
ppMatchDetails	[in] Pointer to pointer to NMMatchDetails that receives pointer to created match details. Can be NULL.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* or *templ* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NTemplate format.
N_E_INVALID_OPERATION	Identification is already started.

Remarks

If *ppMatchDetails* is not NULL, the received pointer should be susequently passed to `NMIdentifyNext` function to obtain details of the matching.

Finally match details should be deleted using `NMMatchDetailsFree` function.

See Also

NMatcher Module | HNMatcher | NMMatchDetails | `NMMatchDetailsFree` | `NMIdentifyNext` | `NMIdentifyEnd`

Copyright © 2006 Neurotechnologija

6.10.1.8. NMIsRegistered Function

Checks if NMatcher library is registered.

```
NBool N_API NMIsRegistered(void);
```

Return Values

NTrue if library is registered, NFalse otherwise.

See Also

NMatcher Module | NMRegister | NMUnregister

6.10.1.9. NMMatchDetails Structure

Represents details of matching performed with NMMatcher.

```
typedef struct NMMatchDetails_ { } NMMatchDetails;
```

Fields

FingersMatchDetails	Pointer to NfsmMatchDetails containing matching details from NfsMatcher.
Score	Similarity score of two matched NTemplates.

See Also

NMatcher Module

6.10.1.9.1. NMMatchDetails.FingersMatchDetails Field

Pointer to NfsmMatchDetails containing matching details from NfsMatcher.

```
NfsmMatchDetails * FingersMatchDetails;
```

Remarks

The field has value of NULL if first matched NTemplate does not contain fingers template.

See Also

NMMatchDetails | NfsmMatchDetails

6.10.1.9.2. NMMatchDetails.Score Field

Similarity score of two matched NTemplates.

```
NInt Score;
```

See Also

NMMatchDetails

6.10.1.10. NMMatchDetailsFree Function

Deletes NMMatchDetailsFree.

```
void N_API NMMatchDetailsFree(
        NMMatchDetails * pMatchDetails;
);
```

Parameters

pMatchDetails	[in] Pointer to NMMatchDetails to delete.

Remarks

If *pMatchDetails* is NULL, does nothing.

See Also

NMatcher Module | NMMatchDetails

6.10.1.11. NMRegister Function

Registers NMatcher library.

```
NResult N_API NMRegister(
        const NChar * szServer
);
```

Parameters

szServer	[in] Pointer to string that specifies license manager server name or IP address.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szServer* is NULL.
N_E_LM_CONNECTION	There was error during connection or communication with license manager server.
N_E_LM_NO_MORE_LICENSE	Registered license count achieved license manager license limit for NMatcher library.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using `NMUnregister` function.

See Also

NMatcher Module | `NMUnregister` | `NMIsRegistered`

6.10.1.12. NMReset Function

Sets default values for all parameters of the specified NMatcher.

```
NResult N_API NMReset(
        HNMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to NMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* is NULL.

See Also

NMatcher Module | HNMatcher

6.10.1.13. NMSetParameter Function

Sets value of the specified parameter of the specified NMatcher.

```
NResult N_API NMSetParameter(
        HNMatcher hMatcher,
        NUShort partId,
        NUInt parameterId,
        const void * pValue
);
```

Parameters

hMatcher	[in] Handle to the NMatcher object. Can be NULL if setting static parameter value.
partId	[in] Identifier of the part of the matcher.
parameterId	[in] Identifier of the parameter to set.
pValue	[in] Pointer to the parameter value to set.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*partId* is invalid. - or - Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*parameterId* specifies a non-static parameter and *hMatcher* is NULL. - or - *pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	Value *pValue* points to is out of range.

Copyright © 2006 Neurotechnologija

Error Code	Condition
N_E_INVALID_OPERATION	*hMatcher* is not NULL and identification is started on the matcher.
N_E_PARAMETER	*parameterId* is invalid.
N_E_PARAMETER_READ_ONLY	*parameterId* specifies read-only parameter.

Remarks

partId can be one of the following:

- NM_PART_FINGERS + any *partId*`NfsmSetParameter` function supports
- NM_PART_NONE

If *partId* is NM_PART_FINGERS + any *partId* `NfsmSetParameter` function supports, *parameterId* must be any `NfsmSetParameter` function supports except for NFSMP_MATCHING_THRESHOLD.

If *partId* is NM_PART_NONE, the following values can be used for *parameterId*:

- NFSMP_MATCHING_THRESHOLD

To learn the type of the parameter pass value obtained with NParameterMakeId macro using N_PC_TYPE_ID code and the parameter id via *parameterId* parameter and pointer to NInt that will receive one of N_TYPE_XXX via *pValue* parameter. *hMatcher* can be NULL in this case.

See Also

NMatcher Module | HNMatcher | `NfsmSetParameter` | `NMGetParameter`

6.10.1.14. NMUnregister Function

Unregisters NMatcher library.

```
void N_API NMUnregister(void);
```

See Also

NMatcher Module | `NMRegister` | `NMIsRegistered`

6.10.1.15. NMVerify Function

Compares two packed NTemplate using the specified NMatcher.

```
NResult N_API NMVerify(
        HNMatcher hMatcher,
        const void * template1,
        NSizeType template1Size,
        const void * template2,
        NSizeType template2Size,
        NMMatchDetails * * ppMatchDetails,
        NInt * pScore
);
```

Parameters

hMatcher	[in] Handle to the NMatcher object.
template1	[in] Pointer to memory buffer containing first packed NTemplate.
template1Size	[in] Size of first packed NTemplate.
template2	[in] Pointer to memory buffer containing second packed NTemplate.
template2Size	[in] Size of second packed NTemplate.
ppMatchDetails	[in] Pointer to pointer to NMMatchDetails that receives pointer to created match details. Can be NULL.
pScore	[out] Pointer to NInt that receives similarity score.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher*, *template1*, *template2*, or *pScore* is NULL.
N_E_END_OF_STREAM	*template1Size* or *template2Size* is less than expected.
N_E_FORMAT	Data in memory buffer *template1* or *template2* points to is inconsistent with NFTemplate format.

Copyright © 2006 Neurotechnologija

Error Code	Condition
N_E_INVALID_OPERATION	Identification is started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see NMP_MATCHING_THRESHOLD and NMSetParameter function), and is greater than or equal to matching threshold otherwise.

If *ppMatchDetails* is not NULL, the received pointer should be examined to obtain details of the matching.

Finally match details should be deleted using NMMatchDetailsFree function.

See Also

NMatcher Module | HNMatcher | NMMatchDetails | NMP_MATCHING_THRESHOLD | NMSetParameter | NMMatchDetailsFree | NMIdentifyStart | NMIdentifyNext | NMIdentifyEnd

6.11. NTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

Import library (Windows): NTemplate.dll.lib.

DLL (Windows): NTemplate.dll.

Shared object (Linux): libNTemplate.so.

Requirements (Windows):

- NCore.dll.
- NFRecord.dll.
- NFTemplate.dll.

Requirements (Linux):

- libNCore.so.
- libNFRecord.so.
- libNFTemplate.so.

Modules

NTemplate	Provides functionality for packing, unpack-

	ing and editing Neurotechnologija Templates (NTemplates).

6.11.1. NTemplate Module

Provides functionality for packing, unpacking and editing Neurotechnologija Templates (NTemplates).

Header file: `NTemplate.h`.

Functions

`NTemplateAddFingers`	Adds an empty fingers template to the NTemplate.
`NTemplateAddFingersCopy`	Adds a copy of the fingers template to the NFTemplate.
`NTemplateAddFingersFromMemory`	Unpacks a fingers template from the specified memory buffer and adds it to the NFTemplate.
`NTemplateCalculateSize`	Calculates the size of a packed NTemplate containing fingers template of the specified size.
`NTemplateCheck`	Checks if format of the packed NTemplate is correct.
`NTemplateClear`	Removes all templates from the NTemplate.
`NTemplateClone`	Creates a copy of the NTemplate.
`NTemplateCreate`	Creates an empty NTemplate.
`NTemplateCreateFromMemory`	Unpacks a NTemplate from the specified memory buffer.
`NTemplateFree`	Deletes the NTemplate. After the object is deleted the specified handle is no longer valid.
`NTemplateGetFingers`	Retrieves the fingers template of the NTemplate.
`NTemplateGetSize`	Calculates packed size of the NTemplate.
`NTemplateInfoDispose`	For internal use.
`NTemplatePack`	Packs packed fingers template as NTemplate

	into the specified memory buffer.
NTemplateRemoveFingers	Removes fingers template from the NTemplate.
NTemplateSaveToMemory	Packs the NFTemplate into the specified memory buffer.
NTemplateUnpack	Unpacks packed fingers template from the packed NTemplate.

Structures

NTemplateInfo	For internal use.

Types

HNTemplate	Handle to NTemplate object.

See Also

NTemplate Library

6.11.1.1. NTemplateAddFingers Function

Adds an empty fingers template to the NTemplate.

```
NResult N_API NTemplateAddFingers(
        HNTemplate hTemplate,
        HNFTemplate * pHFingers
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
pHFingers	[out] Pointer to a HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

Copyright © 2006 Neurotechnologija

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHFingers* is NULL.
N_E_INVALID_OPERATION	NTemplate already contains fingers template. See NTemplateGetFingers function.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

NTemplate Module | HNTemplate | HNFTemplate | NTemplateGetFingers

6.11.1.2. NTemplateAddFingersCopy Function

Adds a copy of the fingers template to the NFTemplate.

```
NResult N_API NTemplateAddFingersCopy(
        HNTemplate hTemplate,
        HNFTemplate hSrcFingers,
        HNFTemplate * pHFingers
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
hSrcFingers	[in] Handle to the NFTemplate object.
pHFingers	[out] Pointer to a HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *hSrcFingers* or *pHFingers* is NULL.
N_E_INVALID_OPERATION	NTemplate already contains fingers tem-

Error Code	Condition
	plate. See NTemplateGetFingers function.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

NTemplate Module | HNTemplate | HNFTemplate | NTemplateGetFingers

6.11.1.3. NTemplateAddFingersFromMemory Function

Unpacks a fingers template from the specified memory buffer and adds it to the NFTemplate.

```
NResult N_API NTemplateAddFingersFromMemory(
        HNTemplate hTemplate,
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        HNFTemplate * pHFingers
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
buffer	[in] Pointer to memory buffer that contains packed NFTemplate.
bufferSize	[in] Size of memory buffer that contains packed NFTemplate.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pHFingers	[out] Pointer to HNFTemplate that receives handle to created NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHFingers*, or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.
N_E_INVALID_OPERATION	NTemplate already contains fingers template. See NTemplateGetFingers function.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

For the list of flags that are supported see NFTemplateCreateFromMemory function.

See Also

NTemplate Module | HNTemplate | HNFTemplate | NTemplateGetFingers | NFTemplateCreateFromMemory

6.11.1.4. NTemplateCalculateSize Function

Calculates the size of a packed NTemplate containing fingers template of the specified size.

```
NResult N_API NTemplateCalculateSize(
        NSizeType fingersTemplateSize,
        NSizeType * pSize
);
```

Parameters

fingersTemplateSize	[in] The size of packed fingers template.
pSize	[out] Pointer to NSizeType that receives calculated size of packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*fingersTemplateSize* is not zero and is less than minimal size of packed fingers template.
N_E_ARGUMENT_NULL	*pSize* is NULL.

Remarks

This is a low-level function and can be changed in future version of the library.

Use NFTemplateCalculateSize function to calculate packed size of a fingers template.

See Also

NFTemplate Module | NFTemplateCalculateSize | NTemplateSaveToMemory

6.11.1.5. NTemplateCheck Function

Checks if format of the packed NTemplate is correct.

```
NResult N_API NTemplateCheck(
        const void * buffer,
        NSizeType bufferSize
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NTemplate.
bufferSize	[in] Size of memory buffer that contains packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.

Error Code	Condition
N_E_FORMAT	Data in memory *buffer* buffer points to is inconsistent with NTemplate format.

See Also

NTemplate Module

6.11.1.6. NTemplateClear Function

Removes all templates from the NTemplate.

```
NResult N_API NTemplateClear(
        HNTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.

See Also

NTemplate Module | HNTemplate

6.11.1.7. NTemplateClone Function

Creates a copy of the NTemplate.

```
NResult N_API NTemplateClone(
        HNTemplate hTemplate,
        HNTemplate * pHClonedTemplate
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
pHClonedTemplate	[out] Pointer to HNTemplate that receives handle to created NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pHClonedTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

See Also

NTemplate Module | HNTemplate | NTemplateFree

6.11.1.8. NTemplateCreate Function

Creates an empty NTemplate.

```
NResult N_API NTemplateCreate(
        HNTemplate * pHTemplate
);
```

Parameters

pHTemplate	[out] Pointer to HNTemplate that receives handle to created NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pHTemplate* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NTemplateFree` function.

See Also

NTemplate Module | HNTemplate | `NTemplateFree`

6.11.1.9. NTemplateCreateFromMemory Function

Unpacks a NTemplate from the specified memory buffer.

```
NResult N_API NTemplateCreateFromMemory(
        const void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NTemplateInfo * pInfo,
        HNTemplate * pHTemplate
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NTemplate.
bufferSize	[in] Size of memory buffer that contains packed NTemplate.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pInfo	[out] For internal use. Must be NULL.
pHTemplate	[out] Pointer to HNTemplate that receives handle to newly created NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pHTemplate* or *buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory buffer *buffer* points to is inconsistent with NFTemplate format.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `NTemplateFree` function.

The following flags are supported:

- Flags supported by `NFTemplateCreateFromMemory` function are applied to fingers template (if) contained in the NTemplate.

See Also

NTemplate Module | HNTemplate | NTemplateInfo | `NTemplateFree` | `NFTemplate-CreateFromMemory` | `NTemplateSaveToMemory`

6.11.1.10. NTemplateFree Function

Deletes the NTemplate. After the object is deleted the specified handle is no longer valid.

```
void N_API NTemplateFree(
        HNTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.

Remarks

If *hTemplate* is NULL, does nothing.

See Also

NTemplate Module | HNTemplate

6.11.1.11. NTemplateGetFingers Function

Retrieves the fingers template of the NTemplate.

```
NResult N_API NTemplateGetFingers(
        HNTemplate hTemplate,
        HNFTemplate * pValue
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
pValue	[out] Points to HNFTemplate that receives handle to NFTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pValue* is NULL.

Remarks

If the NTemplate does not contain fingers template, the returned handle is NULL.

See Also

NTemplate Module | HNTemplate

6.11.1.12. NTemplateGetSize Function

Calculates packed size of the NTemplate.

```
NResult N_API NTemplateGetSize(
        HNTemplate hTemplate,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives calculated size of packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* or *pSize* is NULL.

Remarks

For the list of flags that are supported see `NTemplateSaveToMemory` function.

See Also

NTemplate Module | HNTemplate | `NTemplateSaveToMemory`

6.11.1.13. NTemplatePack Function

Packs packed fingers template as NTemplate into the specified memory buffer.

```
NResult N_API NTemplatePack(
        const void * fingersTemplate,
        NSizeType fingersTemplateSize,
        void * buffer,
        NSizeType bufferSize,
        NSizeType * pSize
);
```

Parameters

fingersTemplate	[in] Pointer to memory buffer that contains packed fingers template.
fingersTemplateSize	[in] Size of memory buffer that contains packed fingers template.
buffer	[out] Pointer to memory buffer to store

	packed NTemplate.
bufferSize	[in] Size of memory buffer to store packed NTemplate.
pSize	[out] Pointer to NSizeType that receives size of packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*bufferSize* is less than size required to store packed NTemplate. - or - *fingersTemplateSize* is less than minimal fingers template size. - or - *fingersTemplateSize* is not equal to size stored in memory buffer *fingersTemplate* points to.
N_E_ARGUMENT_NULL	*fingersTemplate* is NULL and *fingersTemplateSize* is not equal to zero. - or - *buffer* or *pSize* is NULL.
N_E_FORMAT	Data in memory buffer *fingersTemplate* points to is inconsistent with NFTemplate format.

Remarks

bufferSize must not be less than value calculated with `NTemplateCalculateSize` function with the same parameter values.

See Also

NFTemplate Module | NTemplateCalculateSize | NTemplateUnpack

6.11.1.14. NTemplateRemoveFingers Function

Removes fingers template from the NTemplate.

```
NResult N_API NTemplateRemoveFingers(
        HNTemplate hTemplate
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hTemplate* is NULL.

Remarks

If the NTemplate does not contain fingers template, does nothing.

See Also

NFTemplate Module | HNTemplate

6.11.1.15. NTemplateSaveToMemory Function

Packs the NFTemplate into the specified memory buffer.

```
NResult N_API NTemplateSaveToMemory(
        HNTemplate hTemplate,
        void * buffer,
        NSizeType bufferSize,
        NUInt flags,
        NSizeType * pSize
);
```

Parameters

hTemplate	[in] Handle to the NTemplate object.
buffer	[out] Pointer to memory buffer to store packed NTemplate. Can be NULL.
bufferSize	[in] Size of memory buffer to store packed NTemplate.
flags	[in] Bitwise combination of zero or more flags that controls behavior of the function.
pSize	[out] Pointer to NSizeType that receives size of packed NTemplate.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*buffer* is not NULL and *bufferSize* is less than size required to store packed NTemplate.
N_E_ARGUMENT_NULL	*hTemplate* or *pSize* is NULL. - or - *buffer* is NULL and *bufferSize* is not zero.

Remarks

If *buffer* is NULL and *bufferSize* is zero the function only calculates the size of the buffer needed and has the same effect as `NTemplateGetSize` function.

If *buffer* is not NULL, *bufferSize* must not be less than value calculated with `NTemplateGetSize` function.

The following flags are supported:

- Flags supported by `NFTemplateSaveToMemory` function are applied to fingers template (if) contained in the NTemplate.

Copyright © 2006 Neurotechnologija

See Also

NTemplate Module | HNTemplate | NTemplateGetSize | NFTemplateSaveToMemory | NTemplateCreateFromMemory

6.11.1.16. NTemplateUnpack Function

Unpacks packed fingers template from the packed NTemplate.

```
NResult N_API NTemplateUnpack(
        const void * buffer,
        NSizeType bufferSize,
        NByte * pMajorVersion,
        NByte * pMinorVersion,
        NUInt * pSize,
        NByte * pHeaderSize,
        const void * * pFingersTemplate,
        NSizeType * pFingersTemplateSize
);
```

Parameters

buffer	[in] Pointer to memory buffer that contains packed NTemplate.
bufferSize	[in] Size of memory buffer that contains packed NTemplate.
pMajorVersion	[out] Pointer to NByte that receives major version of the packed NTemplate. Can be NULL.
pMinorVersion	[out] Pointer to NByte that receives minor version of the packed NTemplate. Can be NULL.
pSize	[out] Pointer to NUInt that receives size of the packed NTemplate. Can be NULL.
pHeaderSize	[out] Pointer to NByte that receives header size of the packed NTemplate. Can be NULL.
pFingersTemplate	[out] Pointer to void * that receives pointer to packed fingers template contained in the packed NTemplate. Can be NULL.
pFingersTemplateSize	[out] Pointer to NSizeType that receives size of packed fingers template contained in the packed NTemplate. Can be NULL.

Copyright © 2006 Neurotechnologija

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*buffer* is NULL.
N_E_END_OF_STREAM	*bufferSize* is less than expected.
N_E_FORMAT	Data in memory *buffer* buffer points to is inconsistent with NTemplate format.

See Also

NTemplate Module | NTemplatePack

6.12. VFExtractor Library

Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using VeriFinger algorithm.

Import library (Windows): VFExtractor.dll.lib.

DLL (Windows): VFExtractor.dll.

Shared object (Linux): libVFExtractor.so.

Requirements (Windows):

- NCore.dll.
- NFRecord.dll.

Requirements (Linux):

- libNCore.so.
- libNFRecord.so.

Modules

VFExtractor	Provides functionality for extracting Neurotechnologija Finger Records (NFRecords) from fingerprint images using VeriFinger algorithm encapsulated in Neurotechnologija

Copyright © 2006 Neurotechnologija

	Fingerprint Features Extractor VF (VFExtractor) object.

6.12.1. VFExtractor Module

Provides functionality for extracting Neurotechnologija Finger Records (NFRecords) from fingerprint images using VeriFinger algorithm encapsulated in Neurotechnologija Fingerprint Features Extractor VF (VFExtractor) object.

Header file: `VFExtractor.h` (includes `VFExtractorParams.h` and `VFExtractorTypes.h`).

Functions

`VfeCopyParameters`	Copies parameter values from one VFExtractor to another.
`VfeCreate`	Creates a VFExtractor.
`VfeExtract`	Extracts a packed NFRecord from the image using the specified VFExtractor.
`VfeExtractFromImage`	Extracts a packed NFRecord from the NImage using the specified VFExtractor.
`VfeExtractUnpacked`	Extracts a NFRecord from the image using the specified VFExtractor.
`VfeExtractUnpackedFromImage`	Extracts a NFRecord from the NImage using the specified VFExtractor.
`VfeFree`	Deletes the VFExtractor. After the object is deleted the specified handle is no longer valid.
`VfeGetMaxTemplateSize`	Retrieves maximal size of packed NFRecord the specified VFExtractor can extract.
`VfeGetParameter`	Retrieves value of the specified parameter of the specified VFExtractor.
`VfeIsRegistered`	Checks if VFExtractor library is registered.
`VfeRegister`	Registers VFExtractor library.
`VfeReset`	Sets default values for all parameters of the specified VFExtractor.
`VfeSetParameter`	Sets value of the specified parameter of the

	specified VFExtractor.
VfeUnregister	Unregisters VFExtractor library.

Enumerations

VfeReturnedImage	Specifies kind of image returned after extraction using VFExtractor.

Types

HVFExtractor	Handle to VFExtractor object.

Macros

VFE_MODE_ATMEL_FINGERCHIP	The mode for Atmel FingerChip sensor.
VFE_MODE_AUTHENTEC_AES4000	The mode for Authentec AES4000 sensor.
VFE_MODE_AUTHENTEC_AFS2	The mode for Authentec AF-S2 sensor.
VFE_MODE_BIOMETRIKA_FX2000	The mode for Biometrika FX2000 scanner.
VFE_MODE_BMF_BLP100	The mode for BMF BLP100 scanner.
VFE_MODE_CROSSMATCH_VERIFIER 300	The mode for CrossMatch Verifier 300 scanner.
VFE_MODE_DIGITALPERSONA_UARE U	The mode for Digital Persona U.are.U scanner.
VFE_MODE_ETHENTICA	The mode for Ethentica scanner.
VFE_MODE_GENERAL	The general mode.
VFE_MODE_IDENTICATORTECHNOLO GY_DF90	The mode for Identicator Technology DF90 scanner.
VFE_MODE_IDENTIX_TOUCHVIEW	The mode for Identix TouchView scanner.
VFE_MODE_KEYTRONIC_SECUREDES KTOP	The mode for Keytronic Secure Desktop scanner.
VFE_MODE_PRECISEBIOMETRICS_100 CS	The mode for Precise Biometrics 100CS scanner.

VFE_MODE_SECUGEN_HAMSTER	The mode for Secugen Hamster scanner.
VFE_MODE_STMICROELECTRONICS_TOUCHCHIP	The mode for STMicroelectronics Touch-Chip sensor.
VFEP_COPYRIGHT	Identifier specifying library copyright static read-only parameter of type N_TYPE_STRING.
VFEP_MODE	Identifier specifying mode (parameter value set) parameter of type N_TYPE_UINT. Parameter value can be one of the VFE_MODE_XXX.
VFEP_NAME	Identifier specifying library name static read-only parameter of type N_TYPE_STRING.
VFEP_RETURNED_IMAGE	Identifier specifying kind of image returned after extraction parameter of type N_TYPE_INT. Parameter value can be one of the VfeReturnedImage enumeration members.
VFEP_VERSION_HIGH	Identifier specifying high part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
VFEP_VERSION_LOW	Identifier specifying low part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

See Also

VFExtractor Library

6.12.1.1. VfeCopyParameters Function

Copies parameter values from one VFExtractor to another.

```
NResult N_API VfeCopyParameters(
        HVFExtractor hDstExtractor,
        HVFExtractor hSrcExtractor
);
```

Parameters

hDstExtractor	[in] Handle to the destination VFExtractor object.
hSrcExtractor	[in] Handle to the source VFExtractor object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hDstExtractor* or *hSrcExtractor* is NULL.

See Also

VFExtractor Module | HVFExtractor

6.12.1.2. VfeCreate Function

Creates a VFExtractor.

```
NResult N_API VfeCreate(
        HVFExtractor * pHExtractor
);
```

Parameters

pHExtractor	[out] Pointer to HVFExtractor that receives handle to created VFExtractor object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pHExtractor* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using `VfeFree` function.

See Also

VFExtractor Module | HVFExtractor | `VfeFree`

6.12.1.3. VfeExtract Function

Extracts a packed NFRecord from the image using the specified VFExtractor.

```
NResult N_API VfeExtract(
        HVFExtractor hExtractor,
        NUShort width,
        NUShort height,
        NSizeType stride,
        NUShort horzResolution,
        NUShort vertResolution,
        NByte * pixels,
        NFPosition position,
        NFImpressionType impressionType,
        void * buffer,
        NSizeType bufferSize,
        NSizeType * pSize,
        NBool * pIsLowQuality
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object.
width	[in] Specifies image width.
height	[in] Specifies image height.
stride	[in] Specifies length of the image row in bytes.
horzResolution	[in] Specifies horizontal resolution of the image.
vertResolution	[in] Specifies vertical resolution of the image.

Copyright © 2006 Neurotechnologija

pixels	[in, out] Pointer to memory block containing image pixels.
position	[in] Specifies finger position.
impressionType	[in] Specifies impression type.
buffer	[out] Pointer to memory buffer to store extracted packed NFRecord.
bufferSize	[in] Size of memory buffer to store extracted packed NFRecord.
pSize	[out] Pointer to NSizeType that receives size of extracted packed NFRecord.
pIsLowQuality	[out] NTrue if fingerprint image is of low quality, NFalse otherwise.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*stride* is less than *width*. - or - *position* is invalid. - or - *impressionType* is invalid. - or - *bufferSize* is less than needed to store extracted packed NFRecord.
N_E_ARGUMENT_NULL	*hExtractor*, *pixels*, *buffer*, *pSize*, or *pIsLowQuality* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolu-*

Error Code	Condition
	tion is out of supported range.

Remarks

position and *impressionType* are written to extracted NFRecord.

Memory for *buffer* must be allocated before calling the function. VfeGetMaxTemplateSize function should be used to learn the size needed for allocation.

See Also

VFExtractor Module | HVFExtractor | VfeGetMaxTemplateSize | VfeExtractUnpacked | VfeExtractFromImage

6.12.1.4. VfeExtractFromImage Function

Extracts a packed NFRecord from the NImage using the specified VFExtractor.

```
NResult N_API VfeExtractFromImage(
        HVFExtractor hExtractor,
        HNImage hImage,
        NFPosition position,
        NFPosition impressionType,
        void * buffer,
        NSizeType bufferSize,
        NSizeType * pSize,
        NBool * pIsLowQuality
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object.
hImage	[in] Handle to the image.
position	[in] Specifies finger position.
impressionType	[in] Specifies impression type.
buffer	[out] Pointer to memory buffer to store extracted packed NFRecord.
bufferSize	[in] Size of memory buffer to store extracted packed NFRecord.
pSize	[out] Pointer to NSizeType that receives size of extracted packed NFRecord.

pIsLowQuality	[out] NTrue if fingerprint image is of low quality, NFalse otherwise.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*hImage* is invalid. - or - *position* is invalid. - or - *impressionType* is invalid. - or - *bufferSize* is less than needed to store extracted packed NFRecord.
N_E_ARGUMENT_NULL	*hExtractor*, *hImage*, *buffer*, *pSize*, or *pIsLowQuality* is NULL.

Remarks

position and *impressionType* are written to extracted NFRecord.

Memory for *buffer* must be allocated before calling the function. VfeGetMaxTemplateSize function should be used to learn the size needed for allocation.

See Also

VFExtractor Module | HVFExtractor | HNImage | VfeGetMaxTemplateSize | VfeExtract | VfeExtractUnpackedFromImage

6.12.1.5. VfeExtractUnpacked Function

Extracts a NFRecord from the image using the specified VFExtractor.

```
NResult N_API VfeExtractUnpacked(
        HVFExtractor hExtractor,
```

```
        NUShort width,
        NUShort height,
        NSizeType stride,
        NUShort horzResolution,
        NUShort vertResolution,
        NByte * pixels,
        NFPosition position,
        NFPosition impressionType,
        HNFRecord * pHRecord,
        NBool * pIsLowQuality
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object.
width	[in] Specifies image width.
height	[in] Specifies image height.
stride	[in] Specifies length of the image row in bytes.
horzResolution	[in] Specifies horizontal resolution of the image.
vertResolution	[in] Specifies vertical resolution of the image.
pixels	[in, out] Pointer to memory block containing image pixels.
position	[in] Specifies finger position.
impressionType	[in] Specifies impression type.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object containing extracted template.
pIsLowQuality	[out] NTrue if fingerprint image is of low quality, NFalse otherwise.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Copyright © 2006 Neurotechnologija

Error Code	Condition
N_E_ARGUMENT	*stride* is less than *width*. - or - *position* is invalid. - or - *impressionType* is invalid.
N_E_ARGUMENT_NULL	*hExtractor*, *pixels*, *pHRecord*, or *pIsLowQuality* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	*width* or *height* is zero. - or - *horzResolution* or *vertResolution* is out of supported range.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

VFExtractor Module | HVFExtractor | HNFRecord | VfeExtract | VfeExtractUnpackedFromImage

6.12.1.6. VfeExtractUnpackedFromImage Function

Extracts a NFRecord from the NImage using the specified VFExtractor.

```
NResult N_API VfeExtractUnpackedFromImage(
        HVFExtractor hExtractor,
        HNImage hImage,
        NFPosition position,
        NFPosition impressionType,
        HNFRecord * pHRecord,
        NBool * pIsLowQuality
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object.
hImage	[in] Handle to the image.

position	[in] Specifies finger position.
impressionType	[in] Specifies impression type.
pHRecord	[out] Pointer to HNFRecord that receives handle to created NFRecord object containing extracted template.
pIsLowQuality	[out] NTrue if fingerprint image is of low quality, NFalse otherwise.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	*hImage* is invalid. - or - *position* is invalid. - or - *impressionType* is invalid.
N_E_ARGUMENT_NULL	*hExtractor*, *hImage*, *pHRecord*, or *pIsLowQuality* is NULL.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

VFExtractor Module | HVFExtractor | HNImage | HNFRecord | VfeExtractFromImage | VfeExtractUnpacked

6.12.1.7. VfeFree Function

Deletes the VFExtractor. After the object is deleted the specified handle is no longer valid.

```
void N_API VfeFree(
        HVFExtractor hExtractor
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object.

Remarks

If *hExtractor* is NULL, does nothing.

See Also

VFExtractor Module | HVFExtractor

6.12.1.8. VfeGetMaxTemplateSize Function

Retrieves maximal size of packed NFRecord the specified VFExtractor can extract.

```
NResult N_API VfeGetMaxTemplateSize(
        HVFExtractor hExtractor,
        NSizeType * pSize
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object.
pSize	[out] Pointer to NSizeType that receives maximal template size.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hExtractor* or *pSize* is NULL.

See Also

VFExtractor Module | HVFExtractor

6.12.1.9. VfeGetParameter Function

Retrieves value of the specified parameter of the specified VFExtractor.

```
NResult N_API VfeGetParameter(
        HVFExtractor hExtractor,
        NUInt parameterId,
        void * pValue
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object. Can be NULL if retrieving static parameter value.
parameterId	[in] Identifier of the parameter to retrieve.
pValue	[out] Pointer to variable that receives parameter value.

Return Values

If the function succeeds and *parameterId* specifies a N_TYPE_STRING type parameter, and *pValue* is NULL, the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*parameterId* specifies a non-static parameter and *hExtractor* is NULL. - or - *parameterId* specifies a non-N_TYPE_STRING type parameter and *pValue* is NULL.
N_E_PARAMETER	*parameterId* is invalid.

Remarks

The following values can be used for *parameterId*:

- NFEP_COPYRIGHT

- NFEP_MODE
- NFEP_NAME
- NFEP_RETURNED_IMAGE
- NFEP_VERSION_HIGH
- NFEP_VERSION_LOW

To learn the type of the parameter pass value obtained with NParameterMakeId macro using N_PC_TYPE_ID code and the parameter id via *parameterId* parameter and pointer to NInt that will receive one of N_TYPE_XXX via *pValue* parameter. *hExtractor* can be NULL in this case.

See Also

VFExtractor Module | HVFExtractor | VfeSetParameter

6.12.1.10. VfeIsRegistered Function

Checks if VFExtractor library is registered.

```
NBool N_API VfeIsRegistered(void);
```

Return Values

NTrue if library is registered, NFalse otherwise.

See Also

VFExtractor Module | VfeRegister | VfeUnregister

6.12.1.11. VfeRegister Function

Registers VFExtractor library.

```
NResult N_API VfeRegister(
        const NChar * szServer
);
```

Parameters

szServer	[in] Pointer to string that specifies license manager server name or IP address.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szServer* is NULL.
N_E_LM_CONNECTION	There was error during connection or communication with license manager server.
N_E_LM_NO_MORE_LICENSE	Registered license count achieved license manager license limit for VFExtractor library.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using `VfeUnregister` function.

See Also

VFExtractor Module | `VfeUnregister` | `VfeIsRegistered`

6.12.1.12. VfeReset Function

Sets default values for all parameters of the specified VFExtractor.

```
NResult N_API VfeReset(
        HVFExtractor hExtractor
);
```

Parameters

hExtractor	[in] Handle to the VFExtractor object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hExtractor* is NULL.

See Also

VFExtractor Module | HVFExtractor

6.12.1.13. VfeReturnedImage Enumeration

Specifies kind of image returned after extraction using VFExtractor.

```
typedef enum VfeReturnedImage_ { } VfeReturnedImage;
```

Members

`vferiBinarized`	Binarized (filtered) image is written to the image used for extraction.
`vferiNone`	The image used for extraction is left un-changed.
`vferiSkeletonized`	Skeletonized image is written to the image used for extraction.

See Also

VFExtractor Module

6.12.1.14. VfeSetParameter Function

Sets value of the specified parameter of the specified VFExtractor.

```
NResult N_API VfeSetParameter(
        HVFExtractor hExtractor,
        NUInt parameterId,
        const void * pValue
);
```

Parameters

`hExtractor`	[in] Handle to the VFExtractor object. Can be NULL if setting static parameter value.
`parameterId`	[in] Identifier of the parameter to set.
`pValue`	[in] Pointer to the parameter value to set.

Return Values

Copyright © 2006 Neurotechnologija

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pValue* is NULL.
N_E_ARGUMENT_OUT_OF_RANGE	Argument value is out of range.
N_E_PARAMETER	Parameter ID is invalid.

Remarks

The following values can be used for *parameterId*:

- NFEP_MODE
- NFEP_RETURNED_IMAGE

To learn the type of the parameter pass value obtained with NParameterMakeId macro using N_PC_TYPE_ID code and the parameter id via *parameterId* parameter and pointer to NInt that will receive one of N_TYPE_XXX via *pValue* parameter. *hExtractor* can be NULL in this case.

See Also

VFExtractor Module | HVFExtractor | VfeGetParameter

6.12.1.15. VfeUnregister Function

Unregisters VFExtractor library.

```
void N_API VfeUnregister(void);
```

See Also

VFExtractor Module | VfeRegister | VfeIsRegistered

6.13. VFMatcher Library

Provides functionality for comparing Neurotechnologija Finger Records using VeriFinger algorithm.

Import library (Windows): VFMatcher.dll.lib.

DLL (Windows): VFMatcher.dll.

Shared object (Linux): `libVFMatcher.so`.

Requirements (Windows):

- `NCore.dll`.
- `NFRecord.dll`.

Requirements (Linux):

- `libNCore.so`.
- `libNFRecord.so`.

Modules

VFMatcher	Provides functionality for comparing Neurotechnologija Finger Records (NFRecords) using VeriFinger algorithm encapsulated in Neurotechnologija Finger Matcher VF (VFMatcher) object.

6.13.1. VFMatcher Module

Provides functionality for comparing Neurotechnologija Finger Records (NFRecords) using VeriFinger algorithm encapsulated in Neurotechnologija Finger Matcher VF (VFMatcher) object.

Header file: `VFMatcher.h` (includes `VFMatcherParams.h` and `VfmMatchDetails.h`).

Functions

`VfmCopyParameters`	Copies parameter values from one VFMatcher to another.
`VfmCreate`	Creates a VFMatcher.
`VfmFree`	Deletes the VFMatcher. After the object is deleted the specified handle is no longer valid.
`VfmGetParameter`	Retrieves value of the specified parameter of the specified VFMatcher.
`VfmIdentifyEnd`	Ends identification using the specified VFMatcher.

VfmIdentifyNext	Compares the specified packed NFRecord with the one identification was started with using the specified VFMatcher.
VfmIdentifyStart	Starts identification with the specified packed NFRecord using the specified VFMatcher.
VfmIsRegistered	Checks if VFMatcher library is registered.
VfmMatchDetailsFree	Deletes VfmMatchDetails.
VfmRegister	Registers VFMatcher library.
VfmReset	Sets default values for all parameters of the specified VFMatcher.
VfmSetParameter	Sets value of the specified parameter of the specified VFMatcher.
VfmUnregister	Unregisters VFMatcher library.
VfmVerify	Compares two packed NFRecords using the specified VFMatcher.

Structures

VfmMatchDetails	Represents details of matching performed with VFMatcher.

Enumerations

VfmSpeed	Specifies matching speed used in VFMatcher.

Types

HVFMatcher	Handle to VFMatcher object.

See Also

VFMatcher Library

Copyright © 2006 Neurotechnologija

Macros

VFM_MODE_ATMEL_FINGERCHIP	The mode for Atmel FingerChip sensor.
VFM_MODE_AUTHENTEC_AES4000	The mode for Authentec AES4000 sensor.
VFM_MODE_AUTHENTEC_AFS2	The mode for Authentec AF-S2 sensor.
VFM_MODE_BIOMETRIKA_FX2000	The mode for Biometrika FX2000 scanner.
VFM_MODE_BMF_BLP100	The mode for BMF BLP100 scanner.
VFM_MODE_CROSSMATCH_VERIFIER 300	The mode for CrossMatch Verifier 300 scanner.
VFM_MODE_DIGITALPERSONA_UARE U	The mode for Digital Persona U.are.U scanner.
VFM_MODE_ETHENTICA	The mode for Ethentica scanner.
VFM_MODE_GENERAL	The general mode.
VFM_MODE_IDENTICATORTECHNOL OGY_DF90	The mode for Identicator Technology DF90 scanner.
VFM_MODE_IDENTIX_TOUCHVIEW	The mode for Identix TouchView scanner.
VFM_MODE_KEYTRONIC_SECUREDE SKTOP	The mode for Keytronic Secure Desktop scanner.
VFM_MODE_PRECISEBIOMETRICS_10 0CS	The mode for Precise Biometrics 100CS scanner.
VFM_MODE_SECUGEN_HAMSTER	The mode for Secugen Hamster scanner.
VFM_MODE_STMICROELECTRONICS_ TOUCHCHIP	The mode for STMicroelectronics TouchChip sensor.
VFMP_COPYRIGHT	Identifier specifying library copyright static read-only parameter of type N_TYPE_STRING.
VFMP_MATCHING_SPEED	Identifier specifying matching speed parameter of type N_TYPE_INT. Parameter value can be one of the VfmSpeed enumeration members.
VFMP_MATCHING_THRESHOLD	Identifier specifying matching threshold (biggest allowed FAR) parameter of type N_TYPE_INT. Parameter value is equal to -12 * log10(FAR) and must be not less than zero (for example, 48 for FAR = 0.01%).

Copyright © 2006 Neurotechnologija

VFMP_MAXIMAL_ROTATION	Identifier specifying modulus of maximal rotation allowed between two matched NFRecords parameter of type N_TYPE_BYTE. Parameter value is specified in 180/128 degrees units and can not be greater than 128 (+-180 degrees).
VFMP_MODE	Identifier specifying mode (parameter value set) parameter of type N_TYPE_UINT. Parameter value can be one of the VFM_MODE_XXX.
VFMP_NAME	Identifier specifying library name static read-only parameter of type N_TYPE_STRING.
VFMP_VERSION_HIGH	Identifier specifying high part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
VFMP_VERSION_LOW	Identifier specifying low part of library version static read-only parameter of type N_TYPE_UINT. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

6.13.1.1. VfmCopyParameters Function

Copies parameter values from one VFMatcher to another.

```
NResult N_API VfmCopyParameters(
        HVFMatcher hDstMatcher,
        HVFMatcher hSrcMatcher
);
```

Parameters

hDstMatcher	[in] Handle to the destination VFMatcher object.
hSrcMatcher	[in] Handle to the source VFMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hDstMatcher* or *hSrcMatcher* is NULL.
N_E_INVALID_OPERATION	Identification is started on *hDstMatcher*.

See Also

VFMatcher Module | HVFMatcher

6.13.1.2. VfmCreate Function

Creates a VFMatcher.

```
NResult N_API VfmCreate(
        HVFMatcher * pHMatcher
);
```

Parameters

pHMatcher	[out] Pointer to HVFMatcher that receives handle to created VFMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*pHMatcher* is NULL.
N_E_OUT_OF_MEMORY	There was not enough memory.

Remarks

Created object must be deleted using VfmFree function.

See Also

VFMatcher Module | HVFMatcher | VfmFree

6.13.1.3. VfmFree Function

Deletes the VFMatcher. After the object is deleted the specified handle is no longer valid.

```
void N_API VfmFree(
        HVFMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to VFMatcher object.

Remarks

If *hMatcher* is NULL, does nothing.

See Also

VFMatcher Module | HVFMatcher

6.13.1.4. VfmGetParameter Function

Retrieves value of the specified parameter of the specified VFMatcher.

```
NResult N_API VfmGetParameter(
        HVFMatcher hMatcher,
        NUInt parameterId,
        void * pValue
);
```

Parameters

hMatcher	[in] Handle to the VFMatcher object. Can be NULL if retrieving static parameter value.
parameterId	[in] Identifier of the parameter to retrieve.
pValue	[out] Pointer to variable that receives parameter value.

Return Values

If the function succeeds and *parameterId* specifies a N_TYPE_STRING type parameter,

and *pValue* is NULL, the return value is length of the string (not including the NULL-terminator) *pValue* should point to.

If the function succeeds and *pValue* is not NULL, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*parameterId* specifies a non-static parameter and *hMatcher* is NULL. - or - *parameterId* specifies a non-N_TYPE_STRING type parameter and *pValue* is NULL.
N_E_PARAMETER	*parameterId* is invalid.

Remarks

The following values can be used for *parameterId*:

- VFMP_COPYRIGHT
- VFMP_MATCHING_SPEED
- VFMP_MATCHING_THRESHOLD
- VFMP_MAXIMAL_ROTATION
- VFMP_MODE
- VFMP_NAME
- VFMP_VERSION_HIGH
- VFMP_VERSION_LOW

To learn the type of the parameter pass value obtained with NParameterMakeId macro using N_PC_TYPE_ID code and the parameter id via *parameterId* parameter and pointer to NInt that will receive one of N_TYPE_XXX via *pValue* parameter. *hMatcher* can be NULL in this case.

See Also

VFMatcher Module | HVFMatcher | `VfmSetParameter`

6.13.1.5. VfmIdentifyEnd Function

Ends identification using the specified VFMatcher.

```
NResult N_API VfmIdentifyEnd(
        HVFMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to the VFMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* is NULL.
N_E_INVALID_OPERATION	Identification is not started.

See Also

VFMatcher Module | HVFMatcher | VfmIdentifyStart | VfmIdentifyNext

6.13.1.6. VfmIdentifyNext Function

Compares the specified packed NFRecord with the one identification was started with using the specified VFMatcher.

```
NResult N_API VfmIdentifyNext(
        HVFMatcher hMatcher,
        const void * templ,
        NSizeType templSize,
        VfmMatchDetails * pMatchDetails,
        NInt * pScore
);
```

Parameters

hMatcher	[in] Handle to the VFMatcher object.
templ	[in] Pointer to memory buffer containing packed NFRecord.
templSize	[in] Size of packed NFRecord.
pMatchDetails	[in, out] Pointer to VfmMatchDetails that is filled with details of the matching. Can be NULL.
pScore	[out] Pointer to NInt that receives similarity

Copyright © 2006 Neurotechnologija

	score.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher*, *templ*, or *pScore* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Identification is not started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see VFMP_MATCHING_THRESHOLD and VfmSetParameter function), and is greater than or equal to matching threshold otherwise.

If *pMatchDetails* is not NULL it should be a pointer obtained using VfmIdentifyStart function.

See Also

VFMatcher Module | HVFMatcher | VfmMatchDetails | VFMP_MATCHING_THRESHOLD | VfmSetParameter | VfmIdentifyStart | VfmIdentifyEnd

6.13.1.7. VfmIdentifyStart Function

Starts identification with the specified packed NFRecord using the specified VFMatcher.

```
NResult N_API VfmIdentifyStart(
        HVFMatcher hMatcher,
        const void * templ,
        NSizeType templSize,
        VfmMatchDetails * * ppMatchDetails
);
```

Parameters

hMatcher	[in] Handle to the VFMatcher object.
templ	[in] Pointer to memory buffer containing packed NFRecord.
templSize	[in] Size of packed NFRecord.
ppMatchDetails	[in] Pointer to pointer to VfmMatchDetails that receives pointer to created match details. Can be NULL.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* or *templ* is NULL.
N_E_END_OF_STREAM	*templSize* is less than expected.
N_E_FORMAT	Data in memory buffer *templ* points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Identification is already started.

Remarks

If *ppMatchDetails* is not NULL, the received pointer should be susequently passed to `VfmIdentifyNext` function to obtain details of the matching.

Finally match details should be deleted using `VfmMatchDetailsFree` function.

See Also

VFMatcher Module | HVFMatcher | VfmMatchDetails | `NfmMatchDetailsFree` | `VfmIdentifyNext` | `VfmIdentifyEnd`

6.13.1.8. VfmIsRegistered Function

Checks if VFMatcher library is registered.

```
NBool N_API VfmIsRegistered(void);
```

Return Values

NTrue if library is registered, NFalse otherwise.

See Also

VFMatcher Module | `VfmRegister` | `VfmUnregister`

6.13.1.9. VfmMatchDetails Structure

Represents details of matching performed with VFMatcher.

```
typedef struct VfmMatchDetails_ { } VfmMatchDetails;
```

Fields

`MatedMinutiaCount`	Number of mated minutiae in first and second matched NFRecords.
`MatedMinutiae`	Pointer to array of NIndexPair containing pairs of indexes of mated minutiae in first and second matched NFRecords.
`Rotation`	Rotation of second matched NFRecord against the first one.
`Score`	Similarity score of two matched NFRecords.
`TranslationX`	Horizontal translation of second matched NFRecord against the first one.
`TranslationY`	Vertical translation of second matched NFRecord against the first one.

See Also

VFMatcher Module

6.13.1.9.1. VfmMatchDetails.MatedMinutiaCount Field

Number of mated minutiae in first and second matched NFRecords.

```
NInt MatedMinutiaCount;
```

See Also

VfmMatchDetails

6.13.1.9.2. VfmMatchDetails.MatedMinutiae Field

Pointer to array of NIndexPair containing pairs of indexes of mated minutiae in first and second matched NFRecords.

```
NIndexPair * MatedMinutiae;
```

See Also

VfmMatchDetails | NIndexPair

6.13.1.9.3. VfmMatchDetails.Rotation Field

Rotation of second matched NFRecord against the first one.

```
NByte Rotation;
```

Remarks

The rotation is specified in 180/128 degrees units in counterclockwise order.

To eliminate rotation, the second NFRecord minutiae and singular points have to be rotated by the value around center of its minutiae bounding box.

See Also

VfmMatchDetails

6.13.1.9.4. VfmMatchDetails.Score Field

Similarity score of two matched NFRecords.

```
NInt Score;
```

See Also

VfmMatchDetails

6.13.1.9.5. VfmMatchDetails.TranslationX Field

Horizontal translation of second matched NFRecord against the first one.

```
NInt TranslationX;
```

Remarks

To eliminate horizontal translation, the second NFRecord minutiae and singular points have to be shifted right by the value. Note that *Rotation* must be eliminated first.

Copyright © 2006 Neurotechnologija

See Also

VfmMatchDetails | *Rotation*

6.13.1.9.6. VfmMatchDetails.TranslationY Field

Vertical translation of second matched NFRecord against the first one.

```
NInt TranslationY;
```

Remarks

To eliminate horizontal translation, the second NFRecord minutiae and singular points have to be shifted down by the value. Note that *Rotation* must be eliminated first.

See Also

VfmMatchDetails | *Rotation*

6.13.1.10. VfmMatchDetailsFree Function

Deletes VfmMatchDetails.

```
void N_API VfmMatchDetailsFree(
        VfmMatchDetails * pMatchDetails
);
```

Parameters

pMatchDetails	[in] Pointer to VfmMatchDetails to delete.

Remarks

If *pMatchDetails* is NULL, does nothing.

See Also

VFMatcher Module | VfmMatchDetails

6.13.1.11. VfmRegister Function

Registers VFMatcher library.

```
NResult N_API VfmRegister(
        const NChar * szServer
);
```

Parameters

szServer	[in] Pointer to string that specifies license manager server name or IP address.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*szServer* is NULL.
N_E_LM_CONNECTION	There was error during connection or communication with license manager server.
N_E_LM_NO_MORE_LICENSE	Registered license count achieved license manager license limit for VFMatcher library.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using `VfmUnregister` function.

See Also

VFMatcher Module | `VfmUnregister` | `VfmIsRegistered`

6.13.1.12. VfmReset Function

Sets default values for all parameters of the specified VFMatcher.

```
NResult N_API VfmReset(
        HVFMatcher hMatcher
);
```

Parameters

hMatcher	[in] Handle to VFMatcher object.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher* is NULL.

See Also

VFMatcher Module | HVFMatcher

6.13.1.13. VfmSetParameter Function

Sets value of the specified parameter of the specified VFMatcher.

```
NResult N_API VfmSetParameter(
        HVFMatcher hMatcher,
        NUInt parameterId,
        const void * pValue
);
```

Parameters

hMatcher	[in] Handle to the VFMatcher object. Can be NULL if setting static parameter value.
parameterId	[in] Identifier of the parameter to set.
pValue	[in] Pointer to the parameter value to set.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT	Value *pValue* points to is invalid.
N_E_ARGUMENT_NULL	*parameterId* specifies a non-static parameter and *hMatcher* is NULL. - or - *pValue* is NULL.

Error Code	Condition
N_E_ARGUMENT_OUT_OF_RANGE	Value *pValue* points to is out of range.
N_E_INVALID_OPERATION	*hMatcher* is not NULL and identification is started on the matcher.
N_E_PARAMETER	*parameterId* is invalid.
N_E_PARAMETER_READ_ONLY	*parameterId* specifies read-only parameter.

Remarks

The following values can be used for *parameterId*:

- VFMP_MATCHING_SPEED
- VFMP_MATCHING_THRESHOLD
- VFMP_MAXIMAL_ROTATION
- VFMP_MODE

To learn the type of the parameter pass value obtained with NParameterMakeId macro using N_PC_TYPE_ID code and the parameter id via *parameterId* parameter and pointer to NInt that will receive one of N_TYPE_XXX via *pValue* parameter. *hMatcher* can be NULL in this case.

See Also

VFMatcher Module | HVFMatcher | VfmGetParameter

6.13.1.14. VfmSpeed Enumeration

Specifies matching speed used in VFMatcher.

```
typedef enum VfmSpeed_ { } VfmSpeed;
```

Members

vfmsHigh	High matching speed (faster but less accurate).
vfmsLow	Low matching speed (slower but more accurate).

6.13.1.15. VfmUnregister Function

Unregisters VFMatcher library.

```
void N_API VfmUnregister(void);
```

See Also

VFMatcher Module | `VfmRegister` | `VfmIsRegistered`

6.13.1.16. VfmVerify Function

Compares two packed NFRecords using the specified VFMatcher.

```
NResult N_API VfmVerify(
        HVFMatcher hMatcher,
        const void * template1,
        NSizeType template1Size,
        const void * template2,
        NSizeType template2Size,
        VfmMatchDetails * * ppMatchDetalis,
        NInt * pScore
);
```

Parameters

hMatcher	[in] Handle to the VFMatcher object.
template1	[in] Pointer to memory buffer containing first packed NFRecord.
template1Size	[in] Size of first packed NFRecord.
template2	[in] Pointer to memory buffer containing second packed NFRecord.
template2Size	[in] Size of second packed NFRecord.
ppMatchDetails	[in] Pointer to pointer to VfmMatchDetails that receives pointer to created match details. Can be NULL.
pScore	[out] Pointer to NInt that receives similarity score.

Return Values

If the function succeeds, the return value is N_OK.

If the function fails, the return value is one of the following error codes:

Error Code	Condition
N_E_ARGUMENT_NULL	*hMatcher*, *template1*, *template2*, or *pScore* is NULL.
N_E_END_OF_STREAM	*template1Size* or *template2Size* is less than expected.
N_E_FORMAT	Data in memory buffer *template1* or *template2* points to is inconsistent with NFRecord format.
N_E_INVALID_OPERATION	Identification is started.

Remarks

Value received via *pScore* is zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see VFMP_MATCHING_THRESHOLD and VfmSetParameter function), and is greater than or equal to matching threshold otherwise.

If *ppMatchDetails* is not NULL, the received pointer should be examined to obtain details of the matching.

Finally match details should be deleted using VfmMatchDetailsFree function.

See Also

VFMatcher Module | HVFMatcher | VfmMatchDetails | VFMP_MATCHING_THRESHOLD | VfmSetParameter | VfmMatchDetailsFree | VfmIdentifyStart | VfmIdentifyNext | VfmIdentifyEnd

Copyright © 2006 Neurotechnologija

Chapter 7. Reference (.NET)

This chapter contains reference of all libraries included in MegaMatcher SDK for .NET developers.

C# language is used where it is needed to provide code.

Libraries

Neurotec	Provides classes that provide infrastructure for Neurotechnologija components.
Neurotec.Biometrics.ANTemplate	Provides functionality for converting ANSI/NIST-ITL 1-2000 standard files to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.
Neurotec.Biometrics.FMRecord	Provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.
Neurotec.Biometrics.FPScannerMan	Provides functionality for working with scanners.
Neurotec.Biometrics.Gui.NFView	Privides functionality for showing the image. Allows to show, move minutiae over the fingerprint image.
Neurotec.Biometrics.NFExtractor	Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using MegaMatcher algorithm.
Neurotec.Biometrics.NFMatcher	Provides functionality for comparing Neurotechnologija Finger Records using MegaMatcher algorithm.
Neurotec.Biometrics.NFRecord	Provides functionality for packing, unpacking and editing Neurotechnologija Finger Records.
Neurotec.Biometrics.NfsMatcher	Provides functionality for comparing Neurotechnologija Fingers Templates.
Neurotec.Biometrics.NFTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Fingers

	Templates.
Neurotec.Biometrics.NMatcher	Provides functionality for comparing Neurotechnologija Templates.
Neurotec.Biometrics.NTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Templates.
Neurotec.Biometrics.VFExtractor	Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using VeriFinger algorithm.
Neurotec.Biometrics.VFMatcher	Provides functionality for comparing Neurotechnologija Finger Records using VeriFinger algorithm.
Neurotec.Images Pro	Provides classes that enable loading, saving and converting images in various formats.

7.1. Neurotec Library

Provides classes that provide infrastructure for Neurotechnologija components.

DLL: `Neurotec.dll`.

Namespaces

Neurotec	Contains classes that provide infrastructure for Neurotechnologija components.

7.1.1. Neurotec Namespace

Contains classes that provide infrastructure for Neurotechnologija components.

Classes

LicenceManagerException	The exception that is thrown when trying to register a Neurotechnologija library with License Manager server and an error has occurred.
NCore	This class supports internal Neurotechnologija libraries infrastructure and should not be used directly in your code.

NeurotecException	The exception that is thrown when unknown error occurred in one of Neurotechnologija libraries.
NotRegisteredException	The exception that is thrown when using unregistered Neurotechnologija library.
NParameters	This class supports internal Neurotechnologija libraries infrastructure and should not be used directly in your code.
NResult	This class supports internal Neurotechnologija libraries infrastructure and should not be used directly in your code.
ParameterException	The exception that is thrown when parameter code provided to a parameter value get or set method is not valid.
ParameterReadOnlyException	The exception that is thrown when parameter, which code is provided to a parameter value set method, is read-only.

Structures

NIndexPair	Represents pair of indexes.
NRational	Represents a signed rational number.
NURational	Represents an unsigned rational number.

Enumerations

NByteOrder	Specifies byte order.

7.1.1.1. NByteOrder Enumeration

Specifies byte order.

```
public enum NByteOrder
```

Members

Member	Description
BigEndian	Big-endian byte order.
LittleEndian	Little-endian byte order.

7.1.1.2. NIndexPair Structure

Represents pair of indexes.

Constructors

NIndexPair	Initializes a new instance of the NIndexPair structure.

Properties

Index1	Gets or sets first index of this NIndexPair.
Index2	Gets or sets second index of this NIndex-Pair.

7.1.1.2.1. Index1 Property

Gets or sets first index of this NIndexPair.

```
public int Index1 {get; set;}
```

Property value

First index of this NIndexPair.

7.1.1.2.2. Index2 Property

Gets or sets second index of this NIndexPair.

```
public int Index2 {get; set;}
```

Property value

Second index of this NIndexPair.

7.1.1.2.3. NIndexPair Constructor

```
public NIndexPair(
        int index1,
        int index2
);
```

Parameters

index1	First index of this NIndexPair.
index2	Second index of this NIndexPair.

7.1.1.3. NRational Structure

Represents a signed rational number.

Constructors

NRational	Initializes a new instance of the NRational structure.

Fields

Empty	Represents a NRational that is a null reference.

Properties

Denominator	Sets or retrieves the NRational value Denominator.
Numerator	Sets or retrieves the NRational value Numerator.

7.1.1.3.1. NRational Constructor

Initializes a new instance of the NRational structure.

```
public NRational(
        int numerator,
        int denominator
);
```

Copyright © 2006 Neurotechnologija

Parameters

numerator	Numerator of this NRational.
denominator	Denominator of this NRational.

7.1.1.3.2. Empty Field

Represents a NRational that is a null reference.

```
public static readonly NRational Empty
```

7.1.1.3.3. Denominator Property

Sets or retrieves the NRational value Denominator.

```
public int Denominator {get; set;}
```

Property value

Denominator of this NRational.

7.1.1.3.4. Numerator Property

Sets or retrieves the NRational value Numerator.

```
public int Numerator {get; set;}
```

Property value

Numerator of this NRational.

7.1.1.4. NURational Structure

Represents an unsigned rational number.

Constructors

NURational	Initializes a new instance of the NURational structure.

Fields

Empty	Represents a NURational that is a null refer-

	ence.

Properties

Denominator	Sets or retrieves the NURational value Denominator.
Numerator	Sets or retrieves the NURational value Numerator.

7.1.1.4.1. NURational Constructor

Initializes a new instance of the NURational structure.

```
public NURational(
        int numerator,
        int denominator
);
```

Parameters

numerator	Numerator of this NURational.
denominator	Denominator of this NURational.

7.1.1.4.2. Empty Field

Represents a NURational that is a null reference.

```
public static readonly NURational Empty
```

7.1.1.4.3. Denominator Property

Sets or retrieves the NURational value Denominator.

```
public int Denominator {get; set;}
```

Property value

Denominator of this NURational.

7.1.1.4.4. Numerator Property

Sets or retrieves the NURational value Numerator.

```
public int Numerator {get; set;}
```

Property value

Numerator of this NURational.

7.1.1.5. NeurotecException Class

The exception that is thrown when unknown error occurred in one of Neurotechnologija libraries.

Properties

Code	Gets a error code.
Message	Gets a message that describes the current exception.

7.1.1.5.1. Code Property

Gets a error code.

```
public int Code {get;}
```

Property value

An error code.

7.1.1.5.2. Message Property

Gets a message that describes the current exception.

```
public override string Message {get;}
```

Property value

An error message.

7.2. Neurotec.Biometrics.ANTemplate Library

Provides functionality for converting ANSI/NIST-ITL 1-2000 standard files to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

DLL: `Neurotec.Biometrics.ANTemplate.dll`.

Namespaces

Neurotec.Biometrics	.NET wrapper encapsulates management functions and properties of ANSI/NIST ITL-1-2000 standard template.

7.2.1. Neurotec.Biometrics Namespace

.NET wrapper encapsulates management functions and properties of ANSI/NIST ITL-1-2000 standard template.

Classes

ANTemplate	The class was designed to manipulate template of ANSI/NIST ITL-1-2000 standard.

7.2.1.1. ANTemplate Class

Constructors

ANTemplate Constructor	Initializes a new instance of the `ANTemplate` class.

Methods

Dispose	Releases the resources used by `ANTemplate`.
Save	Writes ANTemplate object to file.
ToNFTemplate	Creates NFTemplate object from `ANTemplate`.
ToNTemplate	Creates NTemplate object from `ANTemplate`.

Constants

DllName	Name of DLL containing unmanaged part of this class.

Copyright © 2006 Neurotechnologija

MaxDimension	For internal use.

7.2.1.1.1. ANTemplate Constructor

Initializes a new instance of the `ANTemplate` class.

7.2.1.1.1.1. ANTemplate(string fileName)

Initializes a new instance of the `ANTemplate` class from file.

```
public ANTemplate(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

7.2.1.1.1.2. ANTemplate(NFRecord nfRecord, string tot, string dai, string ori, string tcn)

Initializes a new instance of the `ANTemplate` class from NFRecord.

```
public ANTemplate(
        NFRecord nfRecord,
        string tot,
        string dai,
        string ori,
        string tcn
);
```

Parameters

nfRecord	The NFRecord object.
tot	Specifies Type Of Transaction.
dai	Specifies Destination Agency Identifier.
ori	Specifies Originating Agency Identifier.
tcn	Specifies Transaction Control Number.

7.2.1.1.1.3. ANTemplate(NFTemplate nfTemplate, string tot, string dai, string ori, string tcn)

Initializes a new instance of the `ANTemplate` class from NFTemplate.

```
public ANTemplate(
        NFTemplate nfTemplate,
        string tot,
        string dai,
        string ori,
        string tcn
);
```

Parameters

nfTemplate	The NFTemplate object.
tot	Specifies Type Of Transaction.
dai	Specifies Destination Agency Identifier.
ori	Specifies Originating Agency Identifier.
tcn	Specifies Transaction Control Number.

7.2.1.1.1.4. ANTemplate(NTemplate nTemplate, string tot, string dai, string ori, string tcn)

Initializes a new instance of the `ANTemplate` class from NTemplate.

```
public ANTemplate(
        NTemplate nTemplate,
        string tot,
        string dai,
        string ori,
        string tcn
);
```

Parameters

nTemplate	The NTemplate object.
tot	Specifies Type Of Transaction.
dai	Specifies Destination Agency Identifier.
ori	Specifies Originating Agency Identifier.
tcn	Specifies Transaction Control Number.

7.2.1.1.2. Dispose Method

Releases the resources used by `ANTemplate`.

```
public void Dispose();
```

7.2.1.1.3. Save Method

Writes ANTemplate object to file.

```
public void Save(string fileName);
```

Parameters

fileName	A string that contains the name of the file.

7.2.1.1.4. ToNFTemplate Method

Creates NFTemplate object from `ANTemplate`.

```
public NFTemplate ToNFTemplate();
```

Return Values

The NFTemplate object.

See Also

NFTemplate

7.2.1.1.5. ToNTemplate Method

Creates NTemplate object from `ANTemplate`.

```
public NTemplate ToNTemplate();
```

Return Values

The NTemplate object.

See Also

NTemplate

7.3. Neurotec.Biometrics.FMRecord Library

Provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004

Copyright © 2006 Neurotechnologija

standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

DLL: `Neurotec.Biometrics.FMRecord.dll`.

Namespaces

Neurotec.Biometrics	Contains classes that provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

7.3.1. Neurotec.Biometrics Namespace

Contains classes that provides functionality for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates to and/or from Neurotechnologija Finger Records, Neurotechnologija Fingers Templates and Neurotechnologija Templates.

Classes

Class	Description
FmrFingerView	Provides methods and properties for editing and converting ANSI INCITS 378-2004 standard finger views of templates (FmrFingerViews) to Neurotechnologija Finger Records (NFRecords).
FmrFingerView.CoreCollection	Represents the collection of FmrCore.
FmrFingerView.DeltaCollection	Represents the collection of FmrDelta.
FmrFingerView.MinutiaCollection	Represents the collection of FmrMinutia.
FmrFinger-View.MinutiaEightNeighboursCollection	Represents the collection of NFMinu-tiaNeighbour.
FmrFinger-View.MinutiaFourNeighboursCollection	Represents the collection of NFMinu-tiaNeighbour.
FMRecord	Provides methods and properties for load-ing, editing, saving and converting ANSI INCITS 378-2004 standard templates (FMRecords) to and/or from Neurotechnolo-gija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).

Copyright © 2006 Neurotechnologija

Class	Description
FMRecord.FingerViewCollection	Represents the collection of FmrFingerView.
FMRecordInfo	For internal use.

Structures

Structure	Description
FmrCore	Core structure of ANSI/INCIST 378 2004 standard template.
FmrDelta	Core structure of ANSI/INCIST 378 2004 standard template.
FmrMinutia	Core structure of ANSI/INCIST 378 2004 standard template.

7.3.1.1. FmrCore Structure

Core structure of ANSI/INCIST 378 2004 standard template.

Constructors

FmrCore Constructor	Initializes a new instance of the `FmrCore` structure.

Properties

Angle	Gets or sets angle of core.
RawAngle	Gets or sets raw angle of core.
X	Gets or sets x coordinate of core.
Y	Gets or sets y coordinate of core.

Methods

ToString	Returns a String that represents the current structure.

7.3.1.1.1. FmrCore Constructor

Initializes a new instance of the `FmrCore` structure.

7.3.1.1.1.1. FmrCore(ushort,ushort)

```
public FmrCore(
        ushort x,
        ushort y
);
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.

7.3.1.1.1.2. FmrCore(ushort,ushort,int)

```
public FmrCore(
        ushort x,
        ushort y,
        int angle
);
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.
angle	The angle of the core.

7.3.1.1.1.3. FmrCore(ushort,ushort,double)

```
public FmrCore(
        ushort x,
        ushort y,
        double angle
);
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.
angle	The angle of the core.

7.3.1.1.2. Angle Property

Angle of this FmrCore.

```
public double Angle {get;}
```

Property value

The angle of the core.

See Also

FmrCore

7.3.1.1.3. RawAngle Property

```
public int RawAngle {get;}
```

Property value

The row angle of the core.

See Also

FmrCore

7.3.1.1.4. X Property

```
public int X {get;}
```

Property value

X coordinate of the core.

See Also

FmrCore

7.3.1.1.5. Y Property

```
public int Y {get;}
```

Property value

X coordinate of the core.

See Also

FmrCore

7.3.1.1.6. ToString Method

Returns a String that represents the current structure.

```
public override string ToString();
```

Return Values

A String that represents the current structure.

7.3.1.2. FmrDelta Structure

Delta structure of ANSI/INCIST 378 2004 standard template.

Constructors

FmrDelta Constructor	Initializes a new instance of the `FmrDelta` structure.

Properties

Angle1	First angle of this FmrDelta.
Angle2	Second angle of this FmrDelta.
Angle3	Third angle of this FmrDelta.
RawAngle1	First raw angle of this FmrDelta.
RawAngle2	Second raw angle of this FmrDelta.
RawAngle3	Third raw angle of this FmrDelta.
X	X coordinate of this FmrDelta.
Y	Y coordinate of this FmrDelta.

Copyright © 2006 Neurotechnologija

Methods

ToString	Returns a String that represents the current structure.

7.3.1.2.1. FmrDelta Constructor

Initializes a new instance of the `FmrDelta` structure.

7.3.1.2.1.1. FmrDelta(ushort,ushort)

```
public FmrDelta(
        ushort x,
        ushort y
);
```

Parameters

x	X coordinate of the delta.
y	Y coordinate of the delta.

7.3.1.2.1.2. FmrDelta(ushort x, ushort y, int angle1, int angle2, int angle3)

```
public FmrDelta(
        ushort x,
        ushort y,
        int angle1,
        int angle2,
        int angle3
);
```

Parameters

x	X coordinate of the delta.
y	Y coordinate of the delta.
angle1	First angle of the delta.
angle2	Second angle of the delta.
angle3	Third angle of the delta.

Copyright © 2006 Neurotechnologija

7.3.1.2.1.3. public FmrDelta(ushort x, ushort y, double angle1, double angle2, double angle3)

```
public FmrDelta(
        ushort x,
        ushort y,
        double angle1,
        double angle2,
        double angle3
);
```

Parameters

x	X coordinate of the delta.
y	Y coordinate of the delta.
angle1	First angle of the delta.
angle2	Second angle of the delta.
angle3	Third angle of the delta.

7.3.1.2.2. Angle1 Property

First angle of this FmrDelta.

```
public double Angle1 {get;}
```

Property value

The first angle of the delta.

See Also

FmrDelta

7.3.1.2.3. Angle2 Property

Second angle of this FmrDelta.

```
public double Angle2 {get;}
```

Property value

The second angle of the delta.

See Also

FmrDelta

7.3.1.2.4. Angle3 Property

Third angle of this FmrDelta.

```
public double Angle3 {get;}
```

Property value

The third angle of the delta.

See Also

FmrDelta

7.3.1.2.5. RawAngle1 Property

First raw angle of this FmrDelta.

```
public int RawAngle1 {get;}
```

Property value

The first raw angle of the delta.

See Also

FmrDelta

7.3.1.2.6. RawAngle2 Property

Second raw angle of this FmrDelta.

```
public int RawAngle2 {get;}
```

Property value

The second raw angle of the delta.

See Also

FmrDelta

7.3.1.2.7. RawAngle3 Property

Third raw angle of this FmrDelta.

```
public int RawAngle3 {get;}
```

Property value

The third raw angle of the delta.

See Also

FmrDelta

7.3.1.2.8. X Property

X coordinate of this FmrDelta.

```
public int X {get;}
```

Property value

X coordinate of the delta.

See Also

FmrDelta

7.3.1.2.9. Y Property

Y coordinate of this FmrDelta.

```
public int Y {get;}
```

Property value

Y coordinate of the delta.

See Also

FmrDelta

7.3.1.2.10. ToString Method

Returns a String that represents the current structure.

```
public override string ToString();
```

Return Values

A String that represents the current structure.

7.3.1.3. FmrFingerView Class

Provides methods and properties for editing and converting ANSI INCITS 378-2004 standard finger views of templates (FmrFingerViews) to Neurotechnologija Finger Records (NFRecords).

Properties

Cores	Gets cores collection.
Deltas	Gets deltas collection.
FingerPosition	Gets or sets finger position.
FingerQuality	Gets or sets fingerprint quality.
HasEightNeighbourRidgeCounts	Gets or sets a value indicating whether FM-Record finger view has ridge counts to eight neighbours of each minutia.
HasFourNeighbourRidgeCounts	Gets or sets a value indicating whether FM-Record finger view has ridge counts to four neighbour of each minutia.
ImpressionType	Gets or sets the impression type of the finger record.
Minutiae	Gets minutiae collection.
MinutiaeEightNeighbours	Gets minutia eight neighbours collection.
MinutiaeFourNeighbours	Gets minutia four neighbours collection.
ViewNumber	Gets or sets view number to `FmrFinger-View` object.

Methods

GetMaxSize	Retrieves the maximal size of a FmrFinger-View in stored FMRecord containing the specified number of minutiae, cores and deltas and specified ridge counts.
ToNFRecord	Creates NFRecord object from FmrFinger-View.

Copyright © 2006 Neurotechnologija

Constants

FlagNistRidgeCounts	For internal use.
FlagProcessAllExtendedData	For internal use.
FlagSkipNeurotecFields	The flag indicating whether fields of Neurotechnologija should be skipped while loading or saving FmrFingerView.
FlagSkipRidgeCounts	The flag indicating whether ridge counts should be skipped while loading or saving FmrFingerView.
FlagSkipSingularPoints	The flag indicating whether singular points (cores and deltas) should be skipped while loading or saving FmrFingerView.
FlagUseNeurotecFields	The flag indicating whether fields of Neurotechnologija should be used while loading or saving FmrFingerView.
MaxCoreCount	The maximum number of cores a FmrFingerView can contain.
MaxDeltaCount	The maximum number of deltas a FmrFingerView can contain.
MaxDimension	The maximum value for x and y coordinates of a minutia, core or delta in a FmrFingerView.
MaxMinutiaCount	The maximum number of minutiae a FmrFingerView can contain.

7.3.1.3.1. Cores Property

Gets cores collection.

```
public FmrFingerView.CoreCollection Cores {get;}
```

Property value

The FmrFingerView.CoreCollection collection.

See Also

FmrFingerView.CoreCollection

7.3.1.3.2. Deltas Property

Gets deltas collection.

```
public FmrFingerView.DeltaCollection Deltas {get;}
```

Property value

The FmrFingerView.DeltaCollection collection.

See Also

FmrFingerView.DeltaCollection

7.3.1.3.3. FingerPosition Property

Gets or sets finger position.

```
public NFPosition FingerPosition {get; set;}
```

Property value

One of the NFPosition values. By default is NFPosition.Unknown.

See Also

NFPosition

7.3.1.3.4. FingerQuality Property

Gets or sets fingerprint quality.

```
public byte FingerQuality {get; set;}
```

Property value

Fingerprint quality.

7.3.1.3.5. HasEightNeighbourRidgeCounts Property

Gets or sets a value indicating whether FMRecord finger view has ridge counts to eight neighbours of each minutia.

```
public bool HasEightNeighbourRidgeCounts {get; set;}
```

Property value

`true` if FMRecord finger view has ridge counts to eight neighbours of each minutia; otherwise, `false`.

7.3.1.3.6. HasFourNeighbourRidgeCounts Property

Gets or sets a value indicating whether FMRecord finger view has ridge counts to four neighbour of each minutia.

```
public bool HasFourNeighbourRidgeCounts {get; set;}
```

Property value

`true` if FMRecord finger view has ridge counts to four neighbours of each minutia; otherwise, `false`.

7.3.1.3.7. ImpressionType Property

Gets or sets the impression type of the finger record.

```
public NFImpressionType ImpressionType {get; set;}
```

Property value

One of the NFImpressionType values. The default is NFImpressionType.LiveScanPlain.

See Also

NFImpressionType

7.3.1.3.8. Minutiae Property

Gets minutiae collection.

```
public FmrFingerView.MinutiaCollection Minutiae {get;}
```

Property value

A FmrFingerView.MinutiaCollection that contains minutiae.

See Also

FmrFingerView.MinutiaCollection

7.3.1.3.9. MinutiaeEightNeighbours Property

Gets minutia eight neighbours collection.

```
public FmrFingerView.MinutiaEightNeighboursCollection MinutiaeEightNeighbours {get;}
```

Property value

A FmrFingerView.MinutiaEightNeighboursCollection that contains minutiae of eight neighbours.

See Also

FmrFingerView.MinutiaEightNeighboursCollection

7.3.1.3.10. MinutiaeFourNeighbours Property

Gets minutia four neighbours collection.

```
public FmrFingerView.MinutiaFourNeighboursCollection MinutiaeFourNeighbours {get;}
```

Property value

A FmrFingerView.MinutiaFourNeighboursCollection that contains minutiae of four neighbours.

See Also

FmrFingerView.MinutiaFourNeighboursCollection

7.3.1.3.11. ViewNumber Property

Gets or sets view number to `FmrFingerView` object.

```
public byte ViewNumber {get; set;}
```

Property value

A view number.

7.3.1.3.12. GetMaxSize Method

7.3.1.3.12.1. GetMaxSize(int,bool,bool,bool,int,bool,int)

Retrieves the maximal size of a FmrFingerView in stored FMRecord containing the specified number of minutiae, cores and deltas and specified ridge counts.

```
public static int GetMaxSize(
        int minutiaCount,
        bool hasFourNeighbourRidgeCounts,
        bool hasEightNeighbourRidgeCounts,
        bool coresHasAngles,
        int coreCount,
        bool deltasHasAngles,
        int deltaCount
);
```

Parameters

minutiaCount	The number of minutiae.
hasFourNeighbourRidgeCounts	Indicates whether FMRecord finger view has "four neighbour ridge counts".
hasEightNeighbourRidgeCounts	Indicates whether FMRecord finger view has "eight neighbour ridge counts".
coresHasAngles	`true` if cores has angles; otherwise, `false`.
coreCount	The number of cores.
deltasHasAngles	`true` if deltas has angles; otherwise, `false`.
deltaCount	The number of deltas.

Return Values

The maximal size of a FmrFingerView.

7.3.1.3.12.2. GetMaxSize(int,bool,bool,bool,int,bool,int,bool,bool,int,int)

For internal use.

```
public static int GetMaxSize(
        int minutiaCount,
        bool hasFourNeighbourRidgeCounts,
        bool hasEightNeighbourRidgeCounts,
        bool coresHasAngles,
        int coreCount,
        bool deltasHasAngles,
        int deltaCount,
        bool hasNeurotecCurvatures,
        bool hasNeurotecGs,
        int neurotecBOWidth,
        int neurotecBOHeight
);
```

7.3.1.3.13. ToNFRecord Methods

Creates NFRecord object from FmrFingerView.

7.3.1.3.13.1. NFRecord ToNFRecord()

Creates NFRecord object from FmrFingerView.

```
public NFRecord ToNFRecord();
```

Return Values

The `NFRecord` object.

See Also

NFRecord

7.3.1.3.13.2. NFRecord ToNFRecord(uint flags)

Creates NFRecord object from FmrFingerView. Behavior is controlled through flags.

```
public NFRecord ToNFRecord(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The `NFRecord` object.

See Also

NFRecord

7.3.1.4. FmrFingerView.CoreCollection Class

Represents the collection of FmrCore.

Properties

Capacity	Gets or sets the number of elements that the `FmrFingerView.CoreCollection` can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the `FmrFinger-View.CoreCollection`.
Clear	Removes all elements from the `Fm-rFingerView.CoreCollection`.
CopyTo	Copies the elements of the `FmrFinger-View.CoreCollection` to an array, starting at a particular array index.
GetEnumerator	Returns an enumerator that can be used to iterate through the CoreCollection.
Insert	Inserts an element into the `FmrFinger-View.CoreCollection` at the specified index.
RemoveAt	Removes the element at the specified index within the `FmrFinger-View.CoreCollection`.

7.3.1.4.1. Capacity Property

Gets or sets the number of elements that the `FmrFingerView.CoreCollection` can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the `FmrFingerView.CoreCollection` can contain.

See Also

Count

7.3.1.4.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection

7.3.1.4.3. FmrFingerView.CoreCollection.Item Property

Gets or sets the member from collection by index.

```
public FmrCore this[
        int index
] {get; set;};
```

Parameters

index	The index of the FmrCore structure in the collection to get or set.

Property value

A FmrCore structure.

See Also

Count | FmrCore

7.3.1.4.4. Add Method

Adds an object to the `FmrFingerView.CoreCollection`.

```
public int Add(
        FmrCore value
);
```

Parameters

value	The FmrCore to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

7.3.1.4.5. Clear Method

Removes all elements from the `FmrFingerView.CoreCollection`.

```
public virtual void Clear()
```

See Also

RemoveAt

7.3.1.4.6. CopyTo Method

Copies the elements of the `FmrFingerView.CoreCollection` to an array, starting at a particular array index.

```
public void CopyTo(
        Array array,
        int index
);
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

7.3.1.4.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the CoreCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the core collection.

7.3.1.4.8. Insert Method

Inserts an element into the `FmrFingerView.CoreCollection` at the specified index.

```
public void Insert(
        int index,
        FmrCore value
);
```

Parameters

index	The zero-based index location where the FmrCore is inserted.

value	The FmrCore to add to the collection.

See Also

Add

7.3.1.4.9. RemoveAt Method

Removes the element at the specified index within the `FmrFinger-View.CoreCollection`.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the FmrCore to remove.

See Also

Clear

7.3.1.5. FmrFingerView.DeltaCollection Class

Represents the collection of FmrDelta.

Properties

Capacity	Gets or sets the number of elements that the FmrFingerView.DeltaCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the FmrFinger-View.DeltaCollection

Clear	Removes all elements from the FmrFinger-View.DeltaCollection
CopyTo	Copies the elements of the `FmrFinger-View.DeltaCollection` to an array, starting at a particular array index.
GetEnumerator	Returns an enumerator that can be used to it-erate through the `DeltaCollection`.
Insert	Inserts an element into the FmrFinger-View.DeltaCollection at the specified index.
RemoveAt	Removes the element at the specified index within the FmrFingerView.DeltaCollection.

7.3.1.5.1. Capacity Property

Gets or sets the number of elements that the FmrFingerView.DeltaCollection can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the `FmrFingerView.DeltaCollection` can contain.

See Also

Count

7.3.1.5.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection

7.3.1.5.3. FmrFingerView.DeltaCollection.Item Property

Gets or sets the member from collection by index.

```
public FmrDelta this[
        int index
] {get; set;}
```

Property value

A FmrDelta structure.

See Also

Count | FmrDelta

7.3.1.5.4. Add Method

Adds an object to the FmrFingerView.DeltaCollection

```
public int Add(
        FmrDelta value
);
```

Parameters

value	The FmrDelta to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

7.3.1.5.5. Clear Method

Removes all elements from the FmrFingerView.DeltaCollection

```
public virtual void Clear();
```

See Also

RemoveAt

7.3.1.5.6. CopyTo Method

Copies the elements of the `FmrFingerView.DeltaCollection` to an array, starting at a particular array index.

```
public void CopyTo(
        Array array,
        int index
);
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

7.3.1.5.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the `DeltaCollection`.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the checked index collection.

7.3.1.5.8. Insert Method

Inserts an element into the FmrFingerView.DeltaCollection at the specified index.

```
public void Insert(
        int index,
        FmrDelta value
);
```

Parameters

index	The zero-based index location where the FmrDelta is inserted.
value	The FmrDelta to add to the collection.

See Also

Add

7.3.1.5.9. RemoveAt Method

Removes the element at the specified index within the FmrFingerView.DeltaCollection.

```
public virtual void RemoveAt(
        int index
);
```

Copyright © 2006 Neurotechnologija

Parameters

index	The zero-based index of the FmrDelta to remove.

See Also

Clear

7.3.1.6. FmrFingerView.MinutiaCollection Class

Represents the collection of FmrMinutia.

Properties

Capacity	Gets or sets the number of elements that the `FmrFingerView.CoreCollection` can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the FmrFingerView.MinutiaCollection
Clear	Removes all elements from the FmrFingerView.MinutiaCollection.
CopyTo	Copies the elements of the `FmrFingerView.MinutiaCollection` to an array, starting at a particular array index.
GetEnumerator	Returns an enumerator that can be used to iterate through the FmrFingerView.MinutiaCollection.
Insert	Inserts an element into the FmrFingerView.MinutiaCollection at the specified index.
RemoveAt	Removes the element at the specified index within the FmrFingerView.MinutiaCollection.

Copyright © 2006 Neurotechnologija

7.3.1.6.1. Capacity Property

Gets or sets the number of elements that the `FmrFingerView.CoreCollection` can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the `FmrFingerView.MinutiaCollection` can contain.

See Also

Count

7.3.1.6.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection

7.3.1.6.3. FmrFingerView.MinutiaCollection.Item Property

Gets or sets the member from collection by index.

```
public FmrMinutia this[
        int index
] {get; set;}
```

Property value

A FmrMinutia structure.

See Also

FmrMinutia | Count

7.3.1.6.4. Add Method

Adds an object to the FmrFingerView.MinutiaCollection

```
public int Add(
        FmrMinutia value
);
```

Parameters

value	The FmrMinutia to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

7.3.1.6.5. Clear Method

Removes all elements from the FmrFingerView.MinutiaCollection.

```
public virtual void Clear();
```

See Also

RemoveAt

7.3.1.6.6. CopyTo Method

Copies the elements of the `FmrFingerView.MinutiaCollection` to an array, starting at a particular array index.

```
public void CopyTo(
        Array array,
        int index
);
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

7.3.1.6.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the FmrFingerView.MinutiaCollection.

Copyright © 2006 Neurotechnologija

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the checked index collection.

7.3.1.6.8. Insert Method

Inserts an element into the FmrFingerView.MinutiaCollection at the specified index.

```
public void Insert(
        int index,
        FmrMinutia value
);
```

Parameters

index	The zero-based index location where the FmrMinutia is inserted.
value	The FmrMinutia to add to the collection.

See Also

Add

7.3.1.6.9. RemoveAt Method

Removes the element at the specified index within the FmrFingerView.MinutiaCollection.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the FmrMinutia to remove.

See Also

Clear

7.3.1.7. FmrFingerView.MinutiaEightNeighboursCollection Class

Represents the collection of NFMinutiaNeighbour.

Properties

Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

GetEnumerator	Returns an enumerator that can be used to iterate through the FmrFingerView.MinutiaEightNeighboursCollection.

7.3.1.7.1. Count Property

Gets the number of items in the collection.

```
public virtual int Count {get;}
```

Property value

The number of items in the collection

7.3.1.7.2. FmrFingerView.MinutiaEightNeighboursCollection.Item Property

Gets or sets the member from collection by index.

7.3.1.7.2.1. NFMinutiaNeighbour[] this[int] {get;}

```
public const NFMinutiaNeighbour[] this[
        int minutiaIndex
] {get;}
```

Parameters

minutiaIndex	The index of the specified minutia.

Property value

An NFMinutiaNeighbour structures array of specified minutia.

See Also

Count

7.3.1.7.2.2. NFMinutiaNeighbour this[int, int] {get; set;}

```
public const NFMinutiaNeighbour this[
        int minutiaIndex,
        int index
] {get;  set;}
```

Parameters

minutiaIndex	The index of the specified minutia.
index	The index of the specified neighbour minutia to retrieve from the collection.

Property value

A NFMinutiaNeighbour structure of specified neighbour minutia.

See Also

Count

7.3.1.7.3. GetEnumerator Method

Returns an enumerator that can be used to iterate through the FmrFingerView.MinutiaEightNeighboursCollection.

```
public virtual IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the minutia eight neighbours collection.

7.3.1.8. FmrFingerView.MinutiaFourNeighboursCollection Class

Represents the collection of NFMinutiaNeighbour.

Properties

Count	Gets the number of items in the collection.

Item	Gets or sets the member from collection by index.

Methods

GetEnumerator	Returns an enumerator that can be used to iterate through the FmrFingerView.MinutiaFourNeighboursCollection.

7.3.1.8.1. Count Property

Gets the number of items in the collection.

```
public virtual int Count {get;}
```

Property value

The number of items in the collection

7.3.1.8.2. FmrFingerView.MinutiaFourNeighboursCollection.Item Property

Gets or sets the member from collection by index.

7.3.1.8.2.1. NFMinutiaNeighbour[] this[int] {get;}

```
public const NFMinutiaNeighbour[] this[
        int minutiaIndex
] {get;}
```

Parameters

minutiaIndex	The index of the specified minutia.

Property value

An NFMinutiaNeighbour structures array of specified minutia.

See Also

Count

7.3.1.8.2.2. NFMinutiaNeighbour this[int, int] {get; set;}

```
public const NFMinutiaNeighbour this[
        int minutiaIndex,
        int index
] {get; set;}
```

Parameters

minutiaIndex	The index of the specified minutia.
index	The index of the specified neighbour minutia to retrieve from the collection.

Property value

A NFMinutiaNeighbour structure of specified neighbour minutia.

See Also

Count

7.3.1.8.3. GetEnumerator Method

Returns an enumerator that can be used to iterate through the FmrFingerView.MinutiaFourNeighboursCollection.

```
public virtual IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the minutia four neighbours collection.

7.3.1.9. FmrMinutia Structure

Minutia structure of ANSI/INCIST 378 2004 standard template.

Constructors

FmrMinutia Constructor	Initializes a new instance of the `FmrMinutia` structure.

Properties

Angle	Gets or sets the angle of the minutia.
Quality	Gets or sets quality of the minutia.
RawAngle	Gets or sets the raw angle of the minutia.
Type	Gets or sets the type of the minutia.
X	Gets or sets x coordinate of the minutia.
Y	Gets or sets y coordinate of the minutia.

Methods

ToString	Returns a String that represents the current structure.

7.3.1.9.1. FmrMinutia Constructor

7.3.1.9.1.1. FmrMinutia(ushort,ushort,NFMinutiaType,byte)

```
public FmrMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        byte angle
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

7.3.1.9.1.2. FmrMinutia(ushort,ushort,NFMinutiaType,double)

```
public FmrMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        double angle
```

```
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

7.3.1.9.1.3. FmrMinutia(ushort,ushort,NFMinutiaType,byte,byte)

```
public FmrMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        byte angle,
        byte quality
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.
quality	The quality of the minutia.

7.3.1.9.1.4. FmrMinutia(ushort,ushort,NFMinutiaType,double,byte)

```
public FmrMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        double angle,
        byte quality
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.
quality	The quality of the minutia.

7.3.1.9.2. Angle Property

Gets or sets the angle of the minutia.

```
public double Angle {get;}
```

Property value

The angle of the minutia.

See Also

FmrMinutia

7.3.1.9.3. Quality Property

Gets or sets quality of the minutia.

```
public byte Quality {get;}
```

Property value

The quality of the minutia.

See Also

FmrMinutia

7.3.1.9.4. RawAngle Property

Gets or sets the raw angle of the minutia.

```
public byte RawAngle {get;}
```

Property value

The raw angle of the minutia.

Copyright © 2006 Neurotechnologija

See Also

FmrMinutia

7.3.1.9.5. Type Property

Gets or sets the type of the minutia.

```
public NFMinutiaType Type {get;}
```

Property value

The type of the minutia.

See Also

FmrMinutia

7.3.1.9.6. X Property

Gets or sets x coordinate of the minutia.

```
public int X {get;}
```

Property value

The x coordinate of the minutia.

See Also

FmrMinutia

7.3.1.9.7. Y Property

Gets or sets y coordinate of the minutia.

```
public int Y {get;}
```

Property value

The y coordinate of the minutia.

See Also

FmrMinutia

7.3.1.9.8. ToString Method

Returns a String that represents the current structure.

```
public override string ToString();
```

Return Values

A String that represents the current structure.

7.3.1.10. FMRecord Class

Provides methods and properties for loading, editing, saving and converting ANSI INCITS 378-2004 standard templates (FMRecords) to and/or from Neurotechnologija Fingers Templates (NFTemplates) and Neurotechnologija Templates (NTemplates).

Constructors

FMRecord Constructor	Initializes a new instance of the `FMRecord` class.

Properties

CaptureEquipmentCompliance	Gets or sets Capture Equipment Compliance from FMRecord object.
CaptureEquipmentId	Gets or sets Capture Equipment Id from FMRecord object.
CbeffProductId	Retrieves CBEFF Product Id from FMRecord object.
FingerViews	Retrieves FMRecord.FingerViewCollection collection.
ResolutionX	Retrieves horizontal resolution of scanned image from FMRecord.
ResolutionY	Retrieves vertical resolution of scanned image from FMRecord object.
SizeX	Retrieves horizontal size of scanned image from FMRecord object.
SizeY	Retrieves vertical size of scanned image from FMRecord object.

Methods

Copyright © 2006 Neurotechnologija

CalculateSize	For internal use.
Clone	Creates `FMRecord` object from another `FMRecord` object.
Dispose	Releases the resources used by FMRecord.
GetSize	Calculates packed size of `FMRecord` object.
Save	Packs FMRecord object to byte array.
ToNFTemplate	Creates NFTemplate object from FMRecord.
ToNTemplate	Creates NTemplate object from FMRecord.

Constants

DllName	Name of DLL containing unmanaged part of this class.
FlagProcessFirstFingerViewOnly	The flag indicating whether only the first finger view should be loaded or saved while loading or saving FMRecord.
MaxFingerViewCount	The maximum number of finger views FMRecord can contain.

7.3.1.10.1. FMRecord Constructor

Initializes a new instance of the `FMRecord` class.

7.3.1.10.1.1. FMRecord(ushort sizeX, ushort sizeY, ushort resolutionX, ushort resolutionY)

```
public FMRecord(
        ushort sizeX,
        ushort sizeY,
        ushort resolutionX,
        ushort resolutionY
);
```

Parameters

sizeX	Specifies horizontal size of fingerprint im-

	age.
sizeY	Specifies vertical size of fingerprint image.
resolutionX	Specifies horizontal resolution of fingerprint image.
resolutionY	Specifies vertical resolution of fingerprint image.

7.3.1.10.1.2. FMRecord(byte[] buffer)

Initializes a new instance of the `FMRecord` class from packed `FMRecord`.

```
public FMRecord(byte[] buffer);
```

Parameters

buffer	The packed `FMRecord`.

7.3.1.10.1.3. FMRecord(byte[] buffer, uint flags)

Initializes a new instance of the `FMRecord` class from packed `FMRecord`. Behavior is controlled through flags.

```
public FMRecord(byte[] buffer, uint flags);
```

Parameters

buffer	The packed `FMRecord`.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

7.3.1.10.1.4. FMRecord(byte[] buffer, out FMRecordInfo info)

Initializes a new instance of the `FMRecord` class from packed `FMRecord`.

```
public FMRecord(byte[] buffer, out FMRecordInfo info);
```

Parameters

buffer	The packed `FMRecord`.

info	The FMRecordInfo object.

7.3.1.10.1.5. FMRecord(byte[] buffer, uint flags, out FMRecordInfo info)

Initializes a new instance of the FMRecord class from packed FMRecord. Behavior is controlled through flags.

```
public FMRecord(
        byte[] buffer,
        uint flags,
        out FMRecordInfo info
);
```

Parameters

buffer	The packed FMRecord.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
info	The FMRecordInfo object.

7.3.1.10.1.6. FMRecord(NFRecord nfRecord)

Initializes a new instance of the FMRecord class from another FMRecord object.

```
public FMRecord(
        NFRecord nfRecord
);
```

Parameters

nfRecord	The NFRecord object.

7.3.1.10.1.7. FMRecord(NFRecord nfRecord, uint flags)

Initializes a new instance of the FMRecord class from another FMRecord object. Behavior is controlled through flags.

```
public FMRecord(
        NFRecord nfRecord,
        uint flags
);
```

Parameters

nfRecord	The NFRecord object.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

7.3.1.10.1.8. FMRecord(NFTemplate nfTemplate)

Initializes a new instance of the `FMRecord` class from the `NFTemplate` object.

```
public FMRecord(
        NFTemplate nfTemplate
);
```

Parameters

nfTemplate	The NFTemplate object.

7.3.1.10.1.9. FMRecord(NFTemplate nfTemplate, uint flags)

Initializes a new instance of the `FMRecord` class from the `NFTemplate` object. Behavior is controlled through flags.

```
public FMRecord(
        NFTemplate nfTemplate,
        uint flags
);
```

Parameters

nfTemplate	The NFTemplate object.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

7.3.1.10.2. CaptureEquipmentCompliance Property

Gets or sets Capture Equipment Compliance from FMRecord object.

```
byte CaptureEquipmentCompliance {get; set;}
```

Property value

The Capture Equipment Compliance.

7.3.1.10.3. CaptureEquipmentId Property

Gets or sets Capture Equipment Id from FMRecord object.

```
ushort CaptureEquipmentId {get; set;}
```

Property value

The Capture Equipment Id.

7.3.1.10.4. CbeffProductId Property

Retrieves CBEFF Product Id from FMRecord object.

```
uint CbeffProductId {get, set;}
```

Property value

The CBEFF.

7.3.1.10.5. FingerViews Property

Retrieves FMRecord.FingerViewCollection collection.

```
public FMRecord.FingerViewCollection FingerViews {get;}
```

Property value

The FMRecord.FingerViewCollection collection.

See Also

FMRecord.FingerViewCollection

7.3.1.10.6. ResolutionX Property

Retrieves horizontal resolution of scanned image from FMRecord.

```
ushort ResolutionX {get;}
```

Property value

The horizontal resolution of scanned image.

See Also

ResolutionY

7.3.1.10.7. ResolutionY Property

Retrieves vertical resolution of scanned image from FMRecord object.

```
ushort ResolutionY {get;}
```

Property value

The vertical resolution of scanned image.

See Also

ResolutionX

7.3.1.10.8. SizeX Property

Retrieves horizontal size of scanned image from FMRecord object.

```
ushort SizeX {get;}
```

Property value

The horizontal size of scanned image.

See Also

SizeY

7.3.1.10.9. SizeY Property

Retrieves vertical size of scanned image from FMRecord object.

```
ushort SizeY {get;}
```

Property value

The vertical size of scanned image.

See Also

SizeX

7.3.1.10.10. Clone Method

Creates `FMRecord` object from another `FMRecord` object.

```
public object Clone();
```

Return Values

See Also

FMRecord Constructor

7.3.1.10.11. Dispose Method

Releases the resources used by FMRecord.

```
public void Dispose();
```

7.3.1.10.12. GetSize Method

Calculates packed size of `FMRecord` object.

7.3.1.10.12.1. GetSize()

```
public int GetSize();
```

Return Values

The packed size of `FMRecord` object.

7.3.1.10.12.2. GetSize(uint flags)

Calculates packed size of `FMRecord` object. Behavior is controlled through flags.

```
public int GetSize(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The packed size of `FMRecord` object.

7.3.1.10.13. Save Method

Packs FMRecord object to byte array.

7.3.1.10.13.1. Save()

Packs FMRecord object to byte array.

```
public byte[] Save();
```

Return Values

The byte array with FMRecord object.

7.3.1.10.13.2. Save(uint flags)

Packs FMRecord object to byte array. Behavior is controlled through flags.

```
public byte[] Save(
                uint flags
        );
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The byte array with FMRecord object.

7.3.1.10.14. ToNFTemplate Methods

Creates NFTemplate object from FMRecord.

7.3.1.10.14.1. ToNFTemplate()

Creates NFTemplate object from FMRecord.

```
public NFTemplate ToNFTemplate();
```

Return Values

The NFTemplate object.

See Also

NFTemplate

7.3.1.10.14.2. ToNFTemplate(uint flags)

Creates NFTemplate object from FMRecord. Behavior is controlled through flags.

```
public NFTemplate ToNFTemplate(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The NFTemplate object.

See Also

NFTemplate

7.3.1.10.15. NTemplate Method

Creates NTemplate object from FMRecord.

7.3.1.10.15.1. ToNTemplate()

Creates NTemplate object from FMRecord.

```
public NTemplate ToNTemplate();
```

Return Values

The NTemplate object.

See Also

NTemplate

7.3.1.10.15.2. ToNTemplate(uint)

Creates NTemplate object from FMRecord. Behavior is controlled through flags.

```
public NTemplate ToNTemplate(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The NTemplate object.

See Also

NTemplate

7.3.1.11. FMRecord.FingerViewCollection Class

Represents the collection of FmrFingerView.

Properties

Capacity	Gets or sets the number of elements that the `FMRecord.FingerViewCollection` can contain.
Item	Gets the member from collection by index.

Methods

Add	Adds an object to the `FMRecord.FingerViewCollection`.
AddCopy	Copies an object to the `FMRecord.FingerViewCollection`

7.3.1.11.1. Capacity Property

Gets or sets the number of elements that the `FMRecord.FingerViewCollection` can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the FMRecord.FingerViewCollection can contain.

Copyright © 2006 Neurotechnologija

7.3.1.11.2. FMRecord.FingerViewCollection.Item Property

Gets the member from collection by index.

```
public FmrFingerView this[
        int index
] {get;};
```

Property value

The FmrFingerView object.

See Also

FmrFingerView

7.3.1.11.3. Add Method

Adds an object to the `FMRecord.FingerViewCollection`.

```
public FmrFingerView Add();
```

Return Values

The FmrFingerView object.

See Also

FmrFingerView

7.3.1.11.4. AddCopy Method

Copies an object to the `FMRecord.FingerViewCollection`

```
public FmrFingerView AddCopy (
        FmrFingerView srcFingerView
);
```

Parameters

srcFingerView	The FmrFingerView object.

Return Values

The FmrFingerView object.

See Also

FmrFingerView

7.4. Neurotec.Biometrics.FPScannerMan Library

Provides functionality for working with scanners.

DLL: `Neurotec.Biometrics.FPScannerMan.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for working with scanners.

7.4.1. Neurotec.Biometrics Namespace

List of MegaMatcher supported scanners under Windows OS could be found in Table 3.3, "Supported scanners for different platforms"

Scanner drivers for Windows OS are available in `install\Fingerprint Scanners` folder.

Classes

Class	Description
FPScanner	One instance represents one physical device.
FPScannerEventArgs	The class contains data of events ScannerAdded, ScannerRemoved
FPScannerImageScannedEventArgs	The class contains data of event ImageScanned.
FPScannerMan	Scanners manager enumerates, creates and monitors scanners.
FPScannerMan.ScannerCollection	Represents the collection of FPScanner

Delegates

Delegate	Description
FPScannerEventHandler	Represents the method that handles the ScannerAdded and ScannerRemoved events.
FPScannerImageScannedEventHandler	Represents the method that handles a Im-

Delegate	Description
	ageScanned event.

7.4.1.1. FPScanner Class

One instance represents one physical device.

Properties

ID	Gets associated device identifier.

Methods

StartCapturing	Starts capturing fingerprint image from certain device.
StartCapturingForOneImage	Starts capturing one fingerprint image from certain device.
StopCapturing	Stops capturing fingerprint image.

Events

FingerPlaced	Occurs when finger is placed on scanner.
FingerRemoved	Occurs when finger is removed from scanner.
ImageScanned	Occurs when image is scanned from scanner.

7.4.1.1.1. ID Property

Gets associated device identifier.

```
public string ID {get;}
```

Property value

An associated device identifier.

7.4.1.1.2. StartCapturing Method

Starts capturing fingerprint image from certain device.

```
public void StartCapturing();
```

See Also

StartCapturingForOneImage | StopCapturing

7.4.1.1.3. StartCapturingForOneImage Method

Starts capturing one fingerprint image from certain device.

```
public void StartCapturingForOneImage();
```

Remarks

Scanner is stoped automatically when fingerprint is scanned.

See Also

StartCapturingForOneImage

7.4.1.1.4. StopCapturing Method

Stops capturing fingerprint image.

```
public void StopCapturing();
```

See Also

StartCapturing | StartCapturingForOneImage

7.4.1.1.5. FingerPlaced Event

Occurs when finger is placed on scanner.

```
public event EventHandler FingerPlaced
```

See Also

FingerRemoved | ImageScanned

7.4.1.1.6. FingerRemoved Event

Occurs when finger is removed from scanner.

```
public event EventHandler FingerRemoved
```

See Also

FingerPlaced | ImageScanned

7.4.1.1.7. ImageScanned Event

Occurs when image is scanned from scanner.

```
public event FPScannerImageScannedEventHandler ImageScanned
```

See Also

FingerPlaced | FingerRemoved | FPScannerImageScannedEventHandler

7.4.1.2. FPScannerEventArgs Class

The class contains data of events ScannerAdded, ScannerRemoved.

Constructors

FPScannerEventArgs Constructor	Initializes a new instance of the `FPScannerEventArgs` class.

Properties

ScannerID	Gets scanner id.

7.4.1.2.1. FPScannerEventArgs Constructor

Initializes a new instance of the `FPScannerEventArgs` class.

```
public FPScannerEventArgs(
        string scannerID
);
```

Parameters

scannerID	A scanner id.

See Also

FPScannerMan | FPScannerEventHandler

7.4.1.2.2. ScannerID Property

Gets scanner id.

```
public string ScannerID {get;}
```

Property value

A string that contains scanner ID.

7.4.1.3. FPScannerImageScannedEventArgs Class

The class contains data of event ImageScanned.

Constructors

FPScannerImageScannedEventArgs Constructor	Initializes a new instance of the `FPScannerImageScannedEventArgs` class.

Properties

Image	Gets scanned image.

7.4.1.3.1. FPScannerImageScannedEventArgs Constructors

Initializes a new instance of the FPScannerImageScannedEventArgs class.

```
public FPScannerImageScannedEventArgs(
        NGrayscaleImage image
);
```

Parameters

image	The NGrayscaleImage image.

See Also

FPScanner | FPScannerImageScannedEventHandler

7.4.1.3.2. Image Property

Gets scanned image.

```
public NGrayscaleImage Image {get;}
```

Property value

The NGrayscaleImage object.

See Also

NGrayscaleImage

7.4.1.4. FPScannerMan Class

Constructors

FPScannerMan Constructor	Initializes a new instance of the FPScannerMan class.

Properties

IsRegistered	Registers ScannerMan library.
Scanners	Gets FPScannerMan.ScannerCollection collection.

Methods

Dispose	Releases the resources used by FPScannerMan.
Register	Registers ScannerMan library.
Unregister	Unregisters ScannerMan library.

Events

ScannerAdded	Occurs when the scanner is connected.
ScannerRemoved	Occurs when the scanner ir unpluged.

Constants

DllName	Name of DLL containing unmanaged part of

	this class.

7.4.1.4.1. FPScannerMan Constructor

Initializes a new instance of the `FPScannerMan` class.

```
public FPScannerMan(
        ISynchronizeInvoke synInvoke
);
```

Parameters

synInvoke	An ISynchronizeInvoke object.

7.4.1.4.2. IsRegistered Property

Registers ScannerMan library.

```
public static bool IsRegistered {get;}
```

Property value

`true` if library is registered, `false` if library is not registered.

See Also

Register | Unregister

7.4.1.4.3. Scanners Property

Gets FPScannerMan.ScannerCollection collection.

```
public FPScannerMan.ScannerCollection Scanners {get;}
```

Property value

A FPScannerMan.ScannerCollection collection.

See Also

FPScannerMan.ScannerCollection

7.4.1.4.4. Dispose Method

Releases the resources used by `FPScannerMan`.

```
public void Dispose();
```

7.4.1.4.5. Register Method

Registers ScannerMan library.

```
public static void Register(
        string server
);
```

Parameters

server	The string that specifies license manager server name or IP address.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

7.4.1.4.6. Unregister Method

Unregisters ScannerMan library.

```
public static void Unregister();
```

See Also

Register | IsRegistered

7.4.1.4.7. ScannerAdded Event

Occurs when the scanner is connected.

```
public event FPScannerEventHandler ScannerAdded
```

See Also

FPScannerEventHandler

7.4.1.4.8. ScannerRemoved Event

Occurs when the scanner ir unpluged.

```
public event FPScannerEventHandler ScannerRemoved
```

See Also

FPScannerEventHandler

7.4.1.5. FPScannerMan.ScannerCollection Class

Represents the collection of FPScanner

Properties

Item	Gets FPScanner from collection by index.

7.4.1.5.1. FPScannerMan.ScannerCollection.Item Property

7.4.1.5.1.1. this[int]

Gets FPScanner from collection by index.

```
public FPScanner this[
                int index
        ] {get;}
```

Property value

A FPScanner object.

See Also

FPScanner

7.4.1.5.1.2. this[string]

Gets FPScanner from collection by scanner ID.

```
public FPScanner this[
        string id
] {get;}
```

Property value

A FPScanner object.

See Also

FPScanner

7.4.1.6. FPScannerEventHandler Delegate

Represents the method that handles the ScannerAdded and ScannerRemoved events.

```
public delegate void FPScannerEventHandler(
        object sender,
        FPScannerEventArgs e
);
```

Parameters

sender	The source of the event.
e	A FPScannerEventArgs that contains the event data.

See Also

FPScannerEventArgs | FPScannerMan | ScannerAdded | ScannerRemoved

7.4.1.7. FPScannerImageScannedEventHandler Delegate

Represents the method that handles a ImageScanned event.

```
public delegate void FPScannerImageScannedEventHandler(
        object sender,
        FPScannerImageScannedEventArgs  ea
);
```

Parameters

sender	The source of the event.
ea	A FPScannerImageScannedEventArgs that contains the event data.

See Also

FPScannerImageScannedEventArgs | FPScanner | ImageScanned

7.5. Neurotec.Biometrics.Gui.NFView Library

Privides functionality for showing the image. Allows to show, move minutiae over the finger-

print image.

DLL: `Neurotec.Biometrics.Gui.NFView.dll`.

Namespaces

Neurotec.Biometrics.Gui	Namespace contains user interface controls.

7.5.1. Fingerprint view component

Namespace contains user interface controls.

The fingerprint view component `NFView` allows a developer to show fingerprint image. The purpose of the interface is lets a programmer manipulate `NFView` control.

Classes

Class	Description
NFView	Initializes the component.

Enumerations

Enumeration	Description
ShownImage	Specifies type of showed image.

7.5.1.1. NFView Class

Constructors

NFView Constructor	Initializes the component.

Properties

AllowHover	Allows or denies mouse hover of minutiae.
AllowSelection	Allows or denies selection of minutiae.
HoveredMinutiaIndex	Gets index of hovered minutia.
Image	Gets or sets fingerprint image.

Copyright © 2006 Neurotechnologija

MinutiaColor	Gets or sets minutiae color. By default is red color.
NeighbourMinutiaColor	Gets or sets neighbour minutiae color. By default orange color.
ResultImage	Bitmap binarized or skeletonized image - the binarized or skeletonized image. Gets or sets fingerprint binarized or skeletonized image.
SelectedMinutiaColor	Gets or sets selected minutiae color. By default magenta color.
SelectedMinutiaIndex	Gets or sets index of selected minutia.
ShownImage	Gets or sets one of None, Original, Result. Original - shows the image, Result - shows the binarized or skeletonized image, None - hides the image.
ShowMinutiae	Tells if minutiae should be displayed. `true` - displays minutiae, `false` - hides minutiae.
Template	Gets or sets the fingerprint template.
Zoom	Gets or sets number greater than zero for zooming the fingerprint view.

Methods

GetMinutiaAtScreenPoint	Gets index of minutia, that displayed at specified screen point.
ScreenPointToMinutiaPosition	Calculates position of minutia from given screen point.

Events

HoveredMinutiaIndexChanged	Occurs when hovered minutia index changed.
ImageChanged	Occurs when image changed.
MinutiaColorChanged	Occurs when minutia color changed.
NeighbourMinutiaColorChanged	Occurs when neighbour minutia color

Copyright © 2006 Neurotechnologija

	changed.
ResultImageChanged	Occurs when result image changed.
SelectedMinutiaColorChanged	Occurs when selected minutia color changed.
SelectedMinutiaIndexChanged	Occurs when selected minutia index changed.
ShownImageChanged	Occurs when shown image changed.
ShownMinutiaeChanged	Occurs when shown minutiae changed.
TemplateChanged	Occurs when template changed.
ZoomChanged	Occurs when zoom changed.

7.5.1.1.1. NFView Constructor

Initializes the component.

```
public NFView();
```

See Also

NFView Class

7.5.1.1.2. AllowHover Property

Allows or denies mouse hover of minutiae.

```
public bool AllowHover {get; set;}
```

Property value

`true` - enable hover, `false` - disable hover.

7.5.1.1.3. AllowSelection Property

Allows or denies selection of minutiae.

```
public bool AllowSelection {get; set;}
```

Property value

`true` - enable selection, `false` - disable selection.

7.5.1.1.4. HoveredMinutiaIndex Property

Gets index of hovered minutia.

```
public int HoveredMinutiaIndex {get;}
```

Property value

Index of hovered minutia.

See Also

SelectedMinutiaIndex

7.5.1.1.5. MinutiaColor Property

Gets or sets neighbour minutiae color. By default orange color.

```
public Color NeighbourMinutiaColor {get; set;}
```

Property value

`System.Drawing.Color` value.

See Also

MinutiaColor, SelectedMinutiaColor

7.5.1.1.6. Image Property

Gets or sets fingerprint image.

```
public Bitmap Image {get; set;}
```

Property value

A `System.Drawing.Bitmap` object.

See Also

ResultImage

7.5.1.1.7. MinutiaColor Property

Gets or sets minutiae color. By default is red color.

```
public Color MinutiaColor {get; set;}
```

Copyright © 2006 Neurotechnologija

Property value

`System.Drawing.Color` value.

See Also

SelectedMinutiaColor, NeighbourMinutiaColor

7.5.1.1.8. ResultImage Property

Bitmap binarized or skeletonized image - the binarized or skeletonized image. Gets or sets fingerprint binarized or skeletonized image.

```
public Bitmap ResultImage {get; set;}
```

Property value

A `System.Drawing.Bitmap` object.

See Also

Image

7.5.1.1.9. MinutiaColor Property

Gets or sets selected minutiae color. By default magenta color.

```
public Color SelectedMinutiaColor {get; set;}
```

Property value

`System.Drawing.Color` value.

See Also

MinutiaColor, NeighbourMinutiaColor

7.5.1.1.10. SelectedMinutiaIndex Property

Gets or sets index of selected minutia.

```
public int SelectedMinutiaIndex {get; set;}
```

Property value

Index of selected minutia.

See Also

Copyright © 2006 Neurotechnologija

HoveredMinutiaIndex

7.5.1.1.11. ShowMinutiae Property

Tells if minutiae should be displayed. `true` - displays minutiae, `false` - hides minutiae.

```
public bool ShowMinutiae {get; set;}
```

Property value

`true` - displays minutiae, `false` - hides minutiae.

See Also

ShownImage

7.5.1.1.12. ShownImage Property

Gets or sets one of None, Original, Result. Original - shows the image, Result - shows the binarized or skeletonized image, None - hides the image.

```
public ShownImage ShownImage {get; set;}
```

Property value

Original - shows the image, Result - shows the binarized or skeletonized image, None - hides the image.

See Also

ShowMinutiae

7.5.1.1.13. Template Property

Gets or sets the fingerprint template.

```
public NFRecord Template {get; set;}
```

Property value

NFRecord object.

7.5.1.1.14. Zoom Property

Gets or sets number greater than zero for zooming the fingerprint view.

```
public float Zoom {get; set;}
```

Property value

Float value. 1.0 - original image size.

7.5.1.1.15. GetMinutiaAtScreenPoint Method

Gets index of minutia, that displayed at specified screen point.

```
public int GetMinutiaAtScreenPoint(int x, int y);
```

Parameters

x, y - position in screen.

Return Values

Index of minutia. -1 if no minutia occurs at given point.

See Also

ScreenPointToMinutiaPosition

7.5.1.1.16. ScreenPointToMintiaPosition Method

Calculates position of minutia from given screen point.

```
public Point ScreenPointToMinutiaPosition(int x, int y);
```

Parameters

x, y - position in screen.

Return Values

Point structure, that contains minutia position.

See Also

GetMinutiaAtScreenPoint

7.5.1.1.17. HoveredMinutiaIndexChanged Event

Occurs when hovered minutia index changed.

```
public event EventHandler HoveredMinutiaIndexChanged;
```

7.5.1.1.18. ImageChanged Event

Copyright © 2006 Neurotechnologija

Occurs when image changed.

```
public event EventHandler ImageChanged;
```

7.5.1.1.19. MinutiaColorChanged Event

Occurs when minutia color changed.

```
public event EventHandler MinutiaColorChanged;
```

7.5.1.1.20. NeighbourMinutiaColorChanged Event

Occurs when neighbour minutia color changed.

```
public event EventHandler NeighbourMinutiaColorChanged;
```

7.5.1.1.21. ResultImageChanged Event

Occurs when result image changed.

```
public event EventHandler ResultImageChanged;
```

7.5.1.1.22. ShownImageChanged Event

Occurs when shown image changed.

```
public event EventHandler ShownImageChanged;
```

7.5.1.1.23. ShownMinutiaeChanged Event

Occurs when shown minutiae changed.

```
public event EventHandler ShownMinutiaeChanged;
```

7.5.1.1.24. SelectedMinutiaColorChanged Event

Occurs when selected minutia color changed.

```
public event EventHandler SelectedMinutiaColorChanged;
```

7.5.1.1.25. SelectedMinutiaIndexChanged Event

Occurs when selected minutia index changed.

```
public event EventHandler SelectedMinutiaIndexChanged;
```

7.5.1.1.26. TemplateChanged Event

Occurs when template changed.

```
public event EventHandler TemplateChanged;
```

7.5.1.1.27. ZoomChanged Event

Occurs when zoom changed.

```
public event EventHandler ZoomChanged;
```

7.5.1.2. ShownImage Enumeration

Specifies type of showed image.

Members

Member	Description
None	An image is hidden.
Original	Original image.
Result	Skeletonized or binarized image.

7.6. Neurotec.Biometrics.NFExtractor Library

Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using MegaMatcher algorithm.

DLL: `Neurotec.Biometrics.NFExtractor.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using MegaMatcher algorithm.

7.6.1. Neurotec.Biometrics Namespace

Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using MegaMatcher algorithm.

Classes

Class	Description
NFExtractor	Provides methods for extracting Neurotechnologija Finger Records (NFRecords) from fingerprint images using MegaMatcher algorithm encapsulated in Neurotechnologija Fingerprint Features Extractor (NFExtractor) object.

Enumerations

Enumeration	Description
NfeReturnedImage	Specifies the returned image.

7.6.1.1. NFExtractor Class

Constructors

NFExtractor Constructor	Initializes a new instance of the `NFExtractor` class.

Properties

ExtractedRidgeCounts	Gets or sets extracted ridge counts.
IsRegistered	Gets a value indicating whether NFExtractor library has been registered.
Mode	Gets or sets scanners mode.
ReturnedImage	Gets or sets the image type.

Methods

CopyParameters	Copies the parameters values from one NFExtractor object to another.
Dispose	Releases the resources used by NFExtractor.
ExtractUnpacked	Extracts features from a fingerprint image

Copyright © 2006 Neurotechnologija

	and do not packs.
Extract	Extracts a NFRecord from a fingerprint image.
GetMaxTemplateSize	Retrieves maximal size of packed NFRecord the specified VFExtractor can extract.
GetParameter	Retrieves parameter by parameter identifier.
GetStaticParameter	Retrieves static parameter by parameter identifier.
Register	Registers Neurotec.Biometrics.NFExtractor library.
Reset	Resets all parameters of NFExtractor object to default values.
SetParameter	Sets the parameter by parameter identifier.
SetStaticParameter	Sets the static parameter by parameter identifier.
Unregister	Unregisters Neurotec.Biometrics.NFExtractor library.
UpdateTemplate	Updates changed NFRecord. Recalculates ridge counts, curvatures and G.

Constants

DllName	Name of DLL containing unmanaged part of this class.
ParameterMode	Identifier specifying mode (parameter value set) parameter of type uint. Parameter value can be one of the ModeXXX.
ModeAtmelFingerchip ModeBiometrikaFX2000 ModeBmfBlp100 ModeCrossmatchVerifier300 ModeDigitalPersonaUareU ModeGeneral ModeIdenticatorTechnologyDF90 ModeIdentixDFR2090 ModeIdentixTouchView ModeKeytronicSecureDesktop	Represents scanners.

ModePreciseBiometrics100CS ModeSecugenHamster ModeSTMicroelectronicsTouchChip	
ParameterCopyright	Identifier specifying library copyright static read-only parameter of type string.
ParameterExtractedRidgeCounts	Identifier specifying type of ridge counts extracted parameter of type int. Parameter value can be one of the NFRidgeCountsType enumeration members.
ParameterName	Identifier specifying library name static read-only parameter of type string.
ParameterReturnedImage	Identifier specifying kind of image returned after extraction parameter of type int. Parameter value can be one of the NfeReturnedImage enumeration members.
ParameterVersionHigh	Identifier specifying high part of library version static read-only parameter of type uint. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
ParameterVersionLow	Identifier specifying low part of library version static read-only parameter of type uint. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

7.6.1.1.1. NFExtractor Constructor

Initializes a new instance of the NFExtractor class.

```
public NFExtractor();
```

7.6.1.1.2. ExtractedRidgeCounts Property

Gets or sets extracted ridge counts.

```
public NFRidgeCountsType ExtractedRidgeCounts {get, set};
```

Property value

One of the NFRidgeCountsType values.

See Also

NFRidgeCountsType

7.6.1.1.3. IsRegistered Property

Gets a value indicating whether NFExtractor library has been registered.

```
public static bool IsRegistered {get}
```

Property value

`true` if Neurotec.Biometrics.NFExtractor library is registered; otherwise, `false`.

See Also

Register | Unregister

7.6.1.1.4. Mode Property

Gets or sets scanners mode.

```
public uint Mode {get, set}
```

Property value

One of the Mode constants.

See Also

Mode

7.6.1.1.5. ReturnedImage Property

Gets or sets the image type.

```
public NfeReturnedImage ReturnedImage {get, set};
```

Property value

One of the NfeReturnedImage value.

See Also

NfeReturnedImage

7.6.1.1.6. CopyParameters Method

Copyright © 2006 Neurotechnologija

Copies the parameters values from one NFExtractor object to another.

```
public static void CopyParameters(
        NFExtractor sourceExtractor,
        NFExtractor destinationExtractor
);
```

Parameters

sourceExtractor	Object of the NFExtractor class.
destinationExtractor	Object of the NFExtractor class.

See Also

Constants

7.6.1.1.7. Dispose Method

Releases the resources used by NFExtractor.

```
public void Dispose();
```

7.6.1.1.8. ExtractUnpacked Method

Extracts features from a fingerprint image and do not packs.

7.6.1.1.8.1. ExtractUnpacked(NGrayscaleImage,NFPosition,NFImpressionType, out bool)

Extracts finger features from NGrayscaleImage object.

```
public NFRecord ExtractUnpacked(
        NGrayscaleImage image,
        NFPosition position,
        NFImpressionType impressionType,
        out bool isLowQuality
);
```

Parameters

image	The NGrayscaleImage image.
position	The finger NFPosition.
impressionType	The finger NFImpressionType.
isLowQuality	`true` if fingerprint image is low quality;

	otherwise, `false`.

Return Values

The NFRecord object.

See Also

Extract | NFRecord

7.6.1.1.8.2. ExtractUnpacked(ushort,ushort,uint,ushort,ushort,IntPtr,NFPosition,NFImpressionType,out bool)

```
public NFRecord ExtractUnpacked(
        ushort width,
        ushort height,
        uint stride,
        ushort horzResolution,
        ushort vertResolution,
        IntPtr pixels,
        NFPosition position,
        NFImpressionType impressionType,
        out bool isLowQuality
);
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
stride	The fingerprint image width, plus the width of the alignment bytes.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.
pixels	The pointer to array of fingerprint image.
position	The finger NFPosition.
impressionType	The finger NFImpressionType.
isLowQuality	`true` if fingerprint image is low quality; otherwise, `false`.

Return Values

The NFRecord object.

See Also

Extract | NFRecord

7.6.1.1.9. Extract Methods

Extracts a NFRecord from a fingerprint image.

7.6.1.1.9.1. Extract(NGrayscaleImage image, NFPosition position, NFImpressionType impressionType, out bool isLowQuality)

Extracts finger features from NGrayscaleImage object.

```
public byte[] Extract(
        NGrayscaleImage image,
        NFPosition position,
        NFImpressionType impressionType,
        bool isLowQuality
        );
```

Parameters

image	The NGrayscaleImage image.
position	The finger NFPosition.
impressionType	The finger NFImpressionType.
isLowQuality	`true` if fingerprint image is low quality; otherwise, `false`.

Return Values

The array of packed NFRecord.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

ExtractUnpacked

7.6.1.1.9.2. Extract(ushort width, ushort height, uint stride, ushort horzResolution, ushort vertResolution, IntPtr image,NFPosition position, NFImpressionType impressionType, out bool isLowQuality)

Extracts finger features from fingerprint image.

```
public byte[] Extract(
        ushort width,
        ushort height,
        uint stride,
        ushort horzResolution,
        ushort vertResolution,
        IntPtr pixels,
        NFPosition position,
        NFImpressionType impressionType,
        out bool isLowQuality
        )
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
stride	The fingerprint image width, plus the width of the alignment bytes.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.
pixels	The pointer to array of fingerprint image.
position	The finger NFPosition.
impressionType	The finger NFImpressionType.
isLowQuality	`true` if fingerprint image is low quality; otherwise, `false`.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

ExtractUnpacked

7.6.1.1.10. GetMaxTemplateSize Method

Retrieves maximal size of packed NFRecord the specified VFExtractor can extract.

```
public int GetMaxTemplateSize();
```

Return Values

Returns maximal template size.

7.6.1.1.11. GetParameter Method

Retrieves parameter by parameter identifier.

```
public object GetParameter(
        ushort parameterId
);
```

Parameters

parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterMode
- ParameterExtractedRidgeCounts
- ParameterReturnedImage

See Also

SetParameter | CopyParameters | Constants

7.6.1.1.12. GetStaticParameter Method

Retrieves static parameter by parameter identifier.

```
public static object GetStaticParameter(
        ushort parameterId
);
```

Parameters

parameterId	The static parameter identifier.

Return Values

The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterCopyright
- ParameterName
- ParameterVersionHigh
- ParameterVersionLow

See Also

SetStaticParameter | CopyParameters | Constants

7.6.1.1.13. Register Method

Registers Neurotec.Biometrics.NFExtractor library.

```
public static void Register(
        string server
);
```

Parameters

server	The string that specifies license manager server name or IP address.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

7.6.1.1.14. Reset Method

Resets all parameters of NFExtractor object to default values.

```
public void Reset();
```

7.6.1.1.15. SetParameter Method

Sets the parameter by parameter identifier.

```
public void SetParameter(
        ushort parameterId,
```

```
        object value
);
```

Parameters

parameterId	The parameter identifier.
value	The parameter value.

See Also

GetParameter | CopyParameters | Constants

7.6.1.1.16. SetStaticParameter Method

Sets the static parameter by parameter identifier.

```
public static void SetStaticParameter(
        ushort parameterId,
        object value
);
```

Parameters

parameterId	The static parameter identifier.
value	The static parameter value.

Remarks

Unused mothod.

See Also

GetStaticParameter | CopyParameters | Constants

7.6.1.1.17. Unregister Method

Unregisters Neurotec.Biometrics.NFExtractor library.

```
public static void Unregister();
```

See Also

Register | IsRegistered

7.6.1.1.18. UpdateTemplate Method

Updates changed NFRecord. Recalculates ridge counts, curvatures and G.

7.6.1.1.18.1. UpdateTemplate(NGrayscaleImage,NFRecord)

```
public void UpdateTemplate(
        NGrayscaleImage image,
        NFRecord record
);
```

Parameters

image	The NGrayscaleImage image.
record	The NFRecord object.

Remarks

This method updates ridge counts and minutiae curvatures and Gs using the image provided.

See Also

NFRecord | NGrayscaleImage

7.6.1.1.18.2. UpdateTemplate(NGrayscaleImage,byte[])

```
public void UpdateTemplate(
        NGrayscaleImage image,
        byte[] templ
);
```

Parameters

image	The NGrayscaleImage image.
templ	The packed NFRecord object.

Remarks

This method updates ridge counts and minutiae curvatures and Gs using the image provided.

See Also

Copyright © 2006 Neurotechnologija

NFRecord | NGrayscaleImage

7.6.1.1.18.3. UpdateTemplate(ushort,ushort,uint,ushort,ushort,IntPtr,NFRecord)

```
public void UpdateTemplate(
        ushort width,
        ushort height,
        uint stride,
        ushort horzResolution,
        ushort vertResolution,
        IntPtr pixels,
        NFRecord record
);
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
stride	The fingerprint image width, plus the width of the alignment bytes.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.
pixels	The pointer to array of fingerprint image.
record	The NFRecord object.

Remarks

This method updates ridge counts and minutiae curvatures and Gs using the image provided.

See Also

NFRecord | NGrayscaleImage

7.6.1.1.18.4. UpdateTemplate(ushort,ushort,uint,ushort,ushort,IntPtr,byte[])

```
public void UpdateTemplate(
        ushort width,
        ushort height,
        uint stride,
        ushort horzResolution,
        ushort vertResolution,
        IntPtr pixels,
```

```
        byte[] templ
);
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
stride	The fingerprint image width, plus the width of the alignment bytes.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.
pixels	The pointer to array of fingerprint image.
templ	The packed NFRecord object.

Remarks

This method updates ridge counts and minutiae curvatures and Gs using the image provided.

See Also

NFRecord | NGrayscaleImage

7.6.1.2. NfeReturnedImage Enumeration

Specifies the returned image.

```
public enum NfeReturnedImage
```

Members

Member name	Description
None	No image.
Binarized	The binarized image.
Skeletonized	The skeletonized image.

7.7. Neurotec.Biometrics.NFMatcher Library

Provides functionality for comparing Neurotechnologija Finger Records using MegaMatcher algorithm.

DLL: `Neurotec.Biometrics.NFMatcher.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for comparing Neurotechnologija Finger Records using MegaMatcher algorithm.

7.7.1. Neurotec.Biometrics Namespace

Provides functionality for comparing Neurotechnologija Finger Records using MegaMatcher algorithm.

Classes

NFMatcher	Provides methods for comparing Neurotechnologija Finger Records (NFRecords) using MegaMatcher algorithm encapsulated in Neurotechnologija Finger Matcher (NFMatcher) object.
NfmMatchDetails	Specifies the matching information.

Enumerations

NfmSpeed	Specifies the matching speed.

7.7.1.1. NFMatcher Class

Provides methods for comparing Neurotechnologija Finger Records (NFRecords) using MegaMatcher algorithm encapsulated in Neurotechnologija Finger Matcher (NFMatcher) object.

Constructors

NFMatcher Constructor	Initializes a new instance of the `NFMatcher` class.

Properties

IsRegistered	Gets a value indicating whether Neurotec.Biometrics.NFMatcher library is registered.
MatchingSpeed	Gets or sets matching speed.
MatchingThreshold	Gets or sets the matching threshold.
MaximalRotation	Gets or sets maximal rotation.
Mode	Gets or sets scanners mode.

Methods

CopyParameters	Copies parameters values from one NFMatcher object to another NFMatcher object.
Dispose	Releases the resources used by NFMatcher.
GetParameter	Gets value of parameter by parameter id.
GetStaticParameter	Gets value of static parameter by parameter id.
IdentifyEnd	Ends identification process.
IdentifyNext	Compares the specified NFRecord with the one identification was started with.
IdentifyStart	Starts identification.
Register	Registers Neurotec.Biometrics.NFMatcher library.
Reset	Resets all NFMatcher parameters to default values.
SetParameter	Sets value of parameter by parameter id.
SetStaticParameter	Sets value of static parameter by parameter id.
Unregister	Unregisters Neurotec.Biometrics.NFMatcher library.
Verify	Compares two NFRecords.

Constants

Copyright © 2006 Neurotechnologija

DllName	Name of DLL containing unmanaged part of this class.
ModeAtmelFingerchip ModeBiometrikaFX2000 ModeBmfBlp100 ModeCrossmatchVerifier300 ModeDigitalPersonaUareU ModeGeneral ModeIdenticatorTechnologyDF90 ModeIdentixDFR2090 ModeIdentixTouchView ModeKeytronicSecureDesktop ModePreciseBiometrics100CS ModeSecugenHamster ModeSTMicroelectronicsTouchChip	Represents scanners.
ParameterCopyright	Identifier specifying library copyright static read-only parameter of type string.
ParameterMatchingSpeed	Identifier specifying matching speed parameter of type int. Parameter value can be one of the NfmSpeed enumeration members.
ParameterMatchingThreshold	Identifier specifying matching threshold (biggest allowed FAR) parameter of type int. Parameter value is equal to - 12 * log10(FAR) and must be not less than zero (for example, 48 for FAR = 0.01%).
ParameterMaximalRotation	Identifier specifying modulus of maximal rotation allowed between two matched NFRecords parameter of type byte. Parameter value is specified in 180/128 degrees units and can not be greater than 128 (180 degrees).
ParameterMode	Identifier specifying mode (parameter value set) parameter of type int. Parameter value can be one of the ModeXXX.
ParameterName	Identifier specifying library name static read-only parameter of type string.
ParameterVersionHigh	Identifier specifying high part of library version static read-only parameter of type int. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.

Copyright © 2006 Neurotechnologija

ParameterVersionLow	Identifier specifying low part of library version static read-only parameter of type int. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

7.7.1.1.1. NFMatcher Constructor

Initializes a new instance of the `NFMatcher` class.

```
public NFMatcher();
```

7.7.1.1.2. IsRegistered Property

Gets a value indicating whether Neurotec.Biometrics.NFMatcher library is registered.

```
public static bool IsRegistered {get;}
```

Property value

`true` if Neurotec.Biometrics.NFMatcher library is registered; otherwise, `false`.

See Also

Register | Unregister

7.7.1.1.3. MatchingSpeed Property

Gets or sets matching speed.

```
public NfmSpeed MatchingSpeed {get; set;}
```

Property value

One of the NfmSpeed values. The default is `NfmSpeed.Low`.

See Also

NfmSpeed

7.7.1.1.4. MatchingThreshold Property

Gets or sets the matching threshold.

```
public int MatchingThreshold {get; set;}
```

Property value

A matching threshold.

See Also

ParameterMatchingThreshold

7.7.1.1.5. MaximalRotation Property

Gets or sets maximal rotation.

```
public byte MaximalRotation {get; set;}
```

Property value

A maximal rotation.

See Also

ParameterMaximalRotation

7.7.1.1.6. Mode Property

Gets or sets scanners mode.

```
public uint Mode {get; set;}
```

Property value

On of the scanner types.

See Also

Constants

7.7.1.1.7. CopyParameters Method

Copies parameters values from one NFMatcher object to another NFMatcher object.

```
public static void CopyParameters(
        NFMatcher sourceMatcher,
        NFMatcher destinationMatcher
);
```

Parameters

sourceMatcher	The NFMatcher object.

destinationMatcher	The NFMatcher object.

See Also

GetParameter | GetStaticParameter | SetParameter | SetStaticParameter

7.7.1.1.8. Dispose Method

Releases the resources used by NFMatcher.

```
public void Dispose();
```

7.7.1.1.9. GetParameter Method

Gets value of parameter by parameter id.

```
public object GetParameter(
        ushort parameterId
);
```

Parameters

parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterMatchingSpeed
- ParameterMatchingThreshold
- ParameterMaximalRotation
- ParameterMode

See Also

SetParameter | CopyParameters | Constants

7.7.1.1.10. GetStaticParameter Method

Gets value of static parameter by parameter id.

Copyright © 2006 Neurotechnologija

```
public static object GetStaticParameter(
        ushort parameterId
);
```

Parameters

parameterId	The static parameter identifier.

Return Values

The parameter value.

Return Values

The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterCopyright
- ParameterName
- ParameterVersionHigh
- ParameterVersionLow

See Also

SetStaticParameter | CopyParameters | Constants

7.7.1.1.11. IdentifyEnd Method

Ends identification process.

```
public void IdentifyEnd();
```

See Also

IdentifyStart | IdentifyNext

7.7.1.1.12. IdentifyNext Method

Compares the specified NFRecord with the one identification was started with.

7.7.1.1.12.1. int IdentifyNext(IntPtr,int)

```
public int IdentifyNext(
```

```
        IntPtr template,
        int templateSize
);
```

Parameters

template	Pointer to the memory block that contains packed NFRecord.
templateSize	The template size.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and returns greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | ParameterMatchingThreshold | NFRecord

7.7.1.1.12.2. IdentifyNext(IntPtr,int,NfmMatchDetails);

```
public int IdentifyNext(
        IntPtr template,
        int templateSize,
        NfmMatchDetails matchDetails
);
```

Parameters

template	Pointer to the memory block that contains packed NFRecord.
templateSize	The template size.
matchDetails	The NfmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and returns greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | ParameterMatchingThreshold | NFRecord

7.7.1.1.12.3. IdentifyNext(byte[])

```
public int IdentifyNext(
        byte[] template
);
```

Parameters

template	The byte array with packed NFRecord.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and returns greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | ParameterMatchingThreshold | NFRecord

7.7.1.1.12.4. IdentifyNext(byte[],NfmMatchDetails)

```
public int IdentifyNext(
        byte[] template,
        NfmMatchDetails matchDetails
);
```

Parameters

template	The byte array with packed NFRecord.
matchDetails	The NfmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and returns greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | ParameterMatchingThreshold | NFRecord

7.7.1.1.13. IdentifyStart Method

Starts identification.

7.7.1.1.13.1. IdentifyStart(IntPtr,int)

```
public void IdentifyStart(
        IntPtr template,
        int templateSize
);
```

Parameters

template	Pointer to the memory block that contains packed NFRecord.
templateSize	The template size.

See Also

IdentifyNext | IdentifyEnd | NFRecord

7.7.1.1.13.2. IdentifyStart(IntPtr,int,out NfmMatchDetails)

```
public void IdentifyStart(
        IntPtr template,
        int templateSize,
        out NfmMatchDetails matchDetails
);
```

Parameters

template	Pointer to the memory block that contains packed NFRecord.
templateSize	The template size.
matchDetails	The NfmMatchDetails object.

See Also

IdentifyNext | IdentifyEnd | NFRecord

7.7.1.1.13.3. IdentifyStart(byte[])

```
public void IdentifyStart(
        byte[] template
);
```

Parameters

template	The byte array with packed NFRecord.

See Also

IdentifyNext | IdentifyEnd | NFRecord

7.7.1.1.13.4. IdentifyStart(byte[], out NfmMatchDetails)

```
public void IdentifyStart(
        byte[] template,
        out NfmMatchDetails matchDetails
);
```

Parameters

template	The byte array with packed NFRecord.
matchDetails	The NfmMatchDetails object.

See Also

IdentifyNext | IdentifyEnd | NFRecord

7.7.1.1.14. Register Method

Registers Neurotec.Biometrics.NFMatcher library.

```
public static void Register(
        string server
);
```

Parameters

server	The string that specifies license manager server name or IP address.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

7.7.1.1.15. Reset Method

Resets all NFMatcher parameters to default values.

```
public void Reset();
```

7.7.1.1.16. SetParameter Method

Sets value of parameter by parameter id.

```
public void SetParameter(
        ushort parameterId,
        object value
);
```

Parameters

parameterId	The parameter identifier.
value	The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterMatchingSpeed
- ParameterMatchingThreshold
- ParameterMaximalRotation
- ParameterMode

See Also

GetParameter | CopyParameters | Constants

7.7.1.1.17. SetStaticParameter Method

Sets value of static parameter by parameter id.

```
public static void SetStaticParameter(
        ushort parameterId,
        object value
);
```

Parameters

parameterId	The static parameter identifier.
value	The static parameter value.

Remarks

Unused method.

See Also

GetStaticParameter | CopyParameters | Constants

7.7.1.1.18. Unregister Method

Unregisters Neurotec.Biometrics.NFMatcher library.

```
public static void Unregister();
```

See Also

Register | IsRegistered

7.7.1.1.19. Verify Method

Compares two NFRecords.

7.7.1.1.19.1. Verify(IntPtr,int,IntPtr,int)

```
public int Verify(
        IntPtr template1,
        int template1Size,
        IntPtr template2,
        int template2Size
);
```

Parameters

template1	Pointer to the memory block that contains packed NFRecord.
template1Size	The template size.
template2	Pointer to the memory block that contains packed NFRecord.
template2Size	The template size.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and returns greater than or equal to matching threshold otherwise.

See Also

NFRecord

7.7.1.1.19.2. Verify(IntPtr,int,IntPtr,int,out NfmMatchDetails)

```
public int Verify(
        IntPtr template1,
        int template1Size,
        IntPtr template2,
        int template2Size,
        out NfmMatchDetails matchDetails
);
```

Parameters

template1	Pointer to the memory block that contains packed NFRecord.

template1Size	The template size.
template2	Pointer to the memory block that contains packed NFRecord.
template2Size	The template size.
matchDetails	The NfmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and returns greater than or equal to matching threshold otherwise.

See Also

NfmMatchDetails | NFRecord

7.7.1.1.19.3. Verify(byte[],byte[])

```
public int Verify(
        byte[] template1,
        byte[] template2
);
```

Parameters

template1	The byte array with packed NFRecord.
template2	The byte array with packed NFRecord.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and returns greater than or equal to matching threshold otherwise.

See Also

NFRecord

7.7.1.1.19.4. Verify(byte[],byte[],out NfmMatchDetails)

```
public int Verify(
        byte[] template1,
        byte[] template2,
        out NfmMatchDetails matchDetails
);
```

Parameters

template1	The byte array with packed NFRecord.
template2	The byte array with packed NFRecord.
matchDetails	The NfmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and returns greater than or equal to matching threshold otherwise.

See Also

NfmMatchDetails | NFRecord

7.7.1.2. NfmMatchDetails Class

Specifies the matching information.

Properties

MatedMinutiae	The NIndexPair structures array.
Score	Specifies score of two matched fingerprints.

Properties

Dispose	Releases the resources used by NfmMatch-Details.

7.7.1.2.1. MatedMinutiae Property

The NIndexPair structures array.

```
public NIndexPair[] MatedMinutiae {get;}
```

Property value

The array of NIndexPair structures.

See Also

NIndexPair

7.7.1.2.2. Score Property

Specifies score of two matched fingerprints.

```
public int Score {get;}
```

Property value

The score of two matched fingerprints.

7.7.1.2.3. Dispose Method

Releases the resources used by NfmMatchDetails.

```
public void Dispose();
```

7.7.1.3. NfmSpeed Enumeration

Specifies the matching speed.

```
public enum NfmSpeed
```

Members

Member	Description
High	The high matching speed.
Low	The low matching speed.

Member	Description
Medium	The medium matching speed.

7.8. Neurotec.Biometrics.NFRecord Library

Provides functionality for packing, unpacking and editing Neurotechnologija Finger Records.

DLL: `Neurotec.Biometrics.NFRecord.dll`.

Namespaces

Neurotec.Biometrics	Provides classes for packing, unpacking and editing Neurotechnologija Finger Records.

7.8.1. Neurotec.Biometrics Namespace

Provides classes for packing, unpacking and editing Neurotechnologija Finger Records.

Classes

NFRecord	Contains functionality for packing, unpacking and editing NFRecord information.
NFRecord.CoreCollection	Represents the collection of NFCore.
NFRecord.DeltaCollection	Represents the collection of NFDelta.
NFRecord.DoubleCoreCollection	Represents the collection of NFDoubleCore.
NFRecord.MinutiaCollection	Represents the collection of NFMinutia.
NFRecord.MinutiaNeighboursCollection	Represents the collection of NFMinutiaNeighbour.

Structures

NFCore	The structure contains information of core.
NFDelta	The structure contains information of delta.
NFDoubleCore	The structure contains information of double core.
NFMinutia	The structure contains information of minu-

	tia.
NFMinutiaNeighbour	The structure contains information of minutia neighbour.

Enumerations

NFImpressionType	Specifies the impression types.
NFMinutiaFormat	Specifies formats of minutia.
NFMinutiaType	Specifies types of minutia.
NFPatternClass	Specifies pattern class of the fingerprint.
NFPosition	Specifies finger position.
NFRidgeCountsType	Specifies type of ridge counts contained in NFRecord.

7.8.1.1. NFRecord Class

Constructors

NFRecord Constructor	Initializes a new instance of the `NFRecord` class.

Properties

Cores	Gets cores collection.
Deltas	Gets deltas collection.
DoubleCores	Gets double cores collection.
G	Gets or sets additional fingerprint coefficient.
Handle	Gets handle to unmanaged `NFRecord` object.
Height	Gets the height of fingerprint image.
HorzResolution	Gets horizontal resolution of fingerprint image.

Copyright © 2006 Neurotechnologija

ImpressionType	Gets or sets the impression type of the finger record.
Minutiae	Gets minutiae collection.
MinutiaeNeighbours	Gets minutia neighbours from NFRecord.
MinutiaFormat	Gets or sets minutia format minutia format from NFRecord.
PatternClass	Gets or sets pattern class.
Position	Gets or sets finger position.
Quality	Gets or sets fingerprint quality.
RidgeCountsType	Gets or sets ridge counts type.
VertResolution	Gets vertical resolution of fingerprint image.
Width	Gets width of fingerprint image.

Methods

Check	Checks if format of packed `NFRecord` is correct.
Clone	Creates `NFRecord` object from another `NFRecord` object.
Dispose	Releases the resources used by NFRecord.
FromHandle	Creates `NFRecord` object from handle.
GetG	Retrieves G from packed `NFRecord`.
GetHeight	Retrieves height of fingerprint image from packed `NFRecord`.
GetHorzResolution	Retrieves horizontal resolution of fingerprint image from packed `NFRecord`.
GetImpressionType	Retrieves impression type from packed NFRecord.
GetMaxSize	Calculates the maximal `NFRecord` size.
GetMaxSizeV1	Calculates the maximal version 1.0 `NFRecord` size.
GetPatternClass	Retrieves fingerprint pattern class from

	packed `NFRecord`.
GetPosition	Retrieves finger position from packed `NFRecord`.
GetQuality	Retrieves fingerprint quality from packed `NFRecord`.
GetSize	Calculates packed size of `NFRecord`.
GetSizeV1	Calculates version 1.0 packed size of NFRecord.
GetVertResolution	Retrieves vertical resolution of fingerprint image from packed `NFRecord`.
GetWidth	Retrieves width of fingerprint image from packed `NFRecord`.
Save	Packs `NFRecord` to byte array.
SaveV1	Packs `NFRecord` to memory buffer in version 1.0 format.

Constants

DllName	Name of DLL containing unmanaged part of this class.
FlagNistRidgeCounts	For internal use.
FlagSaveBlockedOrients	The flag indicating whether blocked orientations should be packed in NFRecord.
FlagSkipBlockedOrients	The flag indicating whether blocked orientations should be skipped while unpacking NFRecord.
FlagSkipCurvatures	The flag indicating whether minutiae curvatures should be skipped while unpacking or packing NFRecord.
FlagSkipGs	The flag indicating whether minutiae gs should be skipped while unpacking or packing NFRecord.
FlagSkipQualities	The flag indicating whether minutiae qualities should be skipped while unpacking or packing NFRecord.

Copyright © 2006 Neurotechnologija

FlagSkipRidgeCounts	The flag indicating whether ridge counts should be skipped while unpacking or packing NFRecord.
FlagSkipSingularPoints	The flag indicating whether singular points (cores, deltas and double cores) should be skipped while unpacking or packing NFRecord.
MaxCoreCount	The maximum number of cores a NFRecord can contain.
MaxDeltaCount	The maximum number of deltas a NFRecord can contain.
MaxDimension	The maximum value for x and y coordinates of a minutia, core, delta or double core in a NFRecord.
MaxDoubleCoreCount	The maximum number of double cores a NFRecord can contain.
MaxMinutiaCount	The maximum number of minutiae a NFRecord can contain.
Resolution	The resolution of minutiae, cores, deltas and double cores coordinates in a NFRecord.

7.8.1.1.1. NFRecord Constructors

Initializes a new instance of the `NFRecord` class.

7.8.1.1.1.1. NFRecord(byte[], uint,out NFRecordInfo)

```
public NFRecord(
        byte[] buffer,
        uint flags,
        out NFRecordInfo info
);
```

Parameters

buffer	The packed `NFRecord` object.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
info	The `NFTemplateInfo` object.

7.8.1.1.1.2. NFRecord(byte[],out NFRecordInfo)

```
public NFRecord(
        byte[] buffer,
        out NFRecordInfo info
);
```

Parameters

buffer	The packed `NFRecord` object.
info	The `NFTemplateInfo` object.

7.8.1.1.1.3. NFRecord(byte[] buffer)

Initializes a new instance of the `NFRecord` class.

```
public NFRecord(
        byte[] buffer
);
```

Parameters

buffer	The packed `NFRecord` object.

7.8.1.1.1.4. NFRecord(byte[] buffer, uint flags)

Initializes a new instance of the `NFRecord` class.

```
public NFRecord(
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	The packed `NFRecord` object.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

7.8.1.1.1.5. NFRecord(ushort,ushort,ushort,ushort)

```
public NFRecord(
        ushort width,
        ushort height,
        ushort horzResolution,
        ushort vertResolution
);
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.

7.8.1.1.1.6. NFRecord(ushort,ushort,ushort,ushort,uint)

```
public NFRecord(
        ushort width,
        ushort height,
        ushort horzResolution,
        ushort vertResolution,
        uint flags
);
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

7.8.1.1.2. Cores Property

Gets cores collection.

```
public NFRecord.CoreCollection Cores {get;}
```

Property value

A NFRecord.CoreCollection that contains cores.

See Also

NFRecord.CoreCollection

7.8.1.1.3. Deltas Property

Gets deltas collection.

```
public NFRecord.DeltaCollection Deltas {get;}
```

Property value

A NFRecord.DeltaCollection that contains deltas.

See Also

NFRecord.DeltaCollection

7.8.1.1.4. DoubleCores Property

```
public NFRecord.DoubleCoreCollection DoubleCores {get;}
```

Property value

A NFRecord.DoubleCoreCollection that contains double cores.

See Also

NFRecord.DoubleCoreCollection

7.8.1.1.5. G Property

Gets or sets additional fingerprint coefficient.

```
public byte G {get; set;}
```

Property value

Fingerprint coefficient.

Remarks

G - average fingerprint ridge density.

7.8.1.1.6. Handle Property

Gets handle to unmanaged `NFRecord` object.

```
public IntPtr Handle {get;}
```

Property value

A handle to unmanaged NFRecord object.

7.8.1.1.7. Height Property

Gets the height of fingerprint image.

```
public ushort Height {get; }
```

Property value

Height of fingerprint image.

See Also

Width

7.8.1.1.8. HorzResolution Property

Gets horizontal resolution of fingerprint image.

```
public ushort HorzResolution {get; }
```

Property value

Horizontal resolution of fingerprint image.

See Also

VertResolution

7.8.1.1.9. ImpressionType Property

Gets or sets the impression type of the finger record.

```
public NFImpressionType ImpressionType {get; set;}
```

Property value

One of the NFImpressionType values. The default is NFImpressionType.LiveScanPlain.

Copyright © 2006 Neurotechnologija

See Also

NFImpressionType

7.8.1.1.10. Minutiae Property

Gets minutiae collection.

```
public NFRecord.MinutiaCollection Minutiae {get;}
```

Property value

A NFRecord.MinutiaCollection that contains minutiae.

See Also

NFRecord.MinutiaCollection

7.8.1.1.11. MinutiaeNeighbours Property

Gets minutia neighbours format from NFRecord.

```
public MinutiaNeighboursCollection MinutiaeNeighbours {get;}
```

Property value

A MinutiaNeighboursCollection collection.

See Also

MinutiaNeighboursCollection

7.8.1.1.12. MinutiaFormat Property

Gets or sets minutia format minutia format from NFRecord.

```
public NFMinutiaFormat MinutiaFormat {get; set;}
```

Property value

One of the NFMinutiaFormat values. The default is NFMinutiaFormat.HasCurvature.

See Also

NFMinutiaFormat

7.8.1.1.13. PatternClass Property

Gets or sets pattern class.

```
public NFPatternClass PatternClass {get; set;}
```

Property value

One of the NFPatternClass values. By default is NFPatternClass.Unknown.

See Also

NFPatternClass

7.8.1.1.14. Position Property

Gets or sets finger position.

```
public NFPosition Position {get; set;}
```

Property value

One of the NFPosition values. By default is NFPosition.Unknown.

See Also

NFPosition

7.8.1.1.15. Quality Property

Gets or sets fingerprint quality.

```
public byte Quality {get; set;}
```

Property value

Fingerprint quality.

7.8.1.1.16. RidgeCountsType Property

Gets or sets ridge counts type.

```
public NFRidgeCountsType RidgeCountsType {get; set;}
```

Property value

One of the NFRidgeCountsType values.

See Also

NFRidgeCountsType

7.8.1.1.17. VertResolution Property

Gets vertical resolution of fingerprint image.

```
public ushort VertResolution {get;}
```

Property value

See Also

HorzResolution

7.8.1.1.18. Width Property

Gets width of fingerprint image.

```
public ushort Width {get;}
```

Property value

Width of fingerprint image.

See Also

Height

7.8.1.1.19. Check Methods

7.8.1.1.19.1. Check(byte[] buffer)

Checks if format of packed `NFRecord` is correct.

```
public static void Check(
        byte[] buffer
);
```

Parameters

buffer	The packed `NFRecord` object.

See Also

Check

7.8.1.1.19.2. Check(IntPtr buffer, int bufferSize)

Checks if format of packed `NFRecord` is correct.

```
public static void Check(
        IntPtr buffer,
        int bufferSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFRecord`.
bufferSize	Size of memory buffer that contains packed `NFRecord`.

See Also

Check

7.8.1.1.20. Clone Method

Creates `NFRecord` object from another `NFRecord` object.

```
public virtual object Clone();
```

Return Values

The new `NFRecord` object.

7.8.1.1.21. Dispose Method

Releases the resources used by NFRecord.

```
public void Dispose();
```

7.8.1.1.22. FromHandle Method

Creates `NFRecord` object from handle.

```
public static NFRecord FromHandle(
        IntPtr handle
);
```

Parameters

handle	Handle to unmanaged NFRecord object.

Return Values

`NFRecord` object.

See Also

NFRecord Constructors

7.8.1.1.23. GetG Method

Retrieves G from packed `NFRecord`.

```
public static byte GetG(
        byte[] buffer
);
```

Parameters

buffer	The packed `NFRecord` object.

Return Values

The G from packed `NFRecord`. G - average fingerprint ridge density.

See Also

G

7.8.1.1.24. GetHeight Method

Retrieves height of fingerprint image from packed NFRecord.

```
public static ushort GetHeight(
        byte[] buffer
);
```

Parameters

buffer	The packed `NFRecord` object.

Return Values

The height of fingerprint image.

See Also

GetWidth

7.8.1.1.25. GetHorzResolution Method

Retrieves horizontal resolution of fingerprint image from packed `NFRecord`.

```
public static ushort GetHorzResolution(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

The horizontal resolution of fingerprint image.

See Also

VertResolution

7.8.1.1.26. GetImpressionType Method

Retrieves impression type from packed NFRecord.

```
public static NFImpressionType GetImpressionType(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

A NFImpressionType enumeration member specifying impression type of fingerprint.

See Also

NFImpressionType | ImpressionType

7.8.1.1.27. GetMaxSize Method

Calculates the maximal `NFRecord` size.

7.8.1.1.27.1. GetMaxSize(NFMinutiaFormat,int,NFRidgeCountsType,int,int,int)

```
public static int GetMaxSize(
                NFMinutiaFormat minutiaFormat,
                int minutiaCount,
                NFRidgeCountsType ridgeCountsType,
                int coreCount,
                int deltaCount,
                int doubleCoreCount
        );
```

Parameters

minutiaFormat	One of the NFMinutiaFormat values.
minutiaCount	The minutiae count.
ridgeCountsType	One of the NFRidgeCountsType values.
coreCount	The cores count.
deltaCount	The deltas count.
doubleCoreCount	The double cores count.

Return Values

The maximal `NFRecord` size. The size depends on method parameters.

7.8.1.1.27.2. GetMaxSize(NFMinutiaFormat,int,NFRidgeCountsType,int,int,int,int,int)

```
public static int GetMaxSize(
                NFMinutiaFormat minutiaFormat,
                int minutiaCount,
                NFRidgeCountsType ridgeCountsType,
                int coreCount,
                int deltaCount,
                int doubleCoreCount,
                int boWidth,
                int boHeight
        );
```

Parameters

minutiaFormat	One of the NFMinutiaFormat values.
minutiaCount	The minutiae count.
ridgeCountsType	One of the NFRidgeCountsType values.
coreCount	The cores count.
deltaCount	The deltas count.
doubleCoreCount	The double cores count.
boWidth	For compatibility with VeriFinger.
boHeight	For compatibility with VeriFinger.

Return Values

The maximal `NFRecord` size. The size depends on method parameters.

7.8.1.1.28. GetMaxSizeV1 Method

Calculates the maximal version 1.0 `NFRecord` size.

7.8.1.1.28.1. GetMaxSizeV1(NFMinutiaFormat,int,int,int,int)

```
public static int GetMaxSizeV1(
        NFMinutiaFormat minutiaFormat,
        int minutiaCount,
        int coreCount,
        int deltaCount,
        int doubleCoreCount
);
```

Parameters

minutiaFormat	One of the NFMinutiaFormat values.
minutiaCount	The minutiae count.
coreCount	The cores count.
deltaCount	The deltas count.
doubleCoreCount	The double cores count.

Return Values

The maximal version 1.0 `NFRecord` size. The size depends on method parameters.

7.8.1.1.28.2. GetMaxSizeV1(NFMinutiaFormat,int,int,int,int,int,int)

```
public static int GetMaxSizeV1(
        NFMinutiaFormat minutiaFormat,
        int minutiaCount,
        int coreCount,
        int deltaCount,
        int doubleCoreCount,
        int boWidth,
        int boHeight
);
```

Parameters

minutiaFormat	One of the NFMinutiaFormat values.
minutiaCount	The minutiae count.
coreCount	The cores count.
deltaCount	The deltas count.
doubleCoreCount	The double cores count.
boWidth	For compatibility with VeriFinger.
boHeight	For compatibility with VeriFinger.

Return Values

The maximal version 1.0 `NFRecord` size. The size depends on method parameters.

7.8.1.1.29. GetPatternClass Method

Retrieves fingerprint pattern class from packed `NFRecord`.

```
public static NFPatternClass GetPatternClass(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

One of the NFPatternClass values.

See Also

PatternClass

7.8.1.1.30. GetPosition Method

Retrieves finger position from packed `NFRecord`.

```
public static NFPosition GetPosition(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

One of the NFPosition values.

See Also

NFPosition | Position

7.8.1.1.31. GetQuality Method

Retrieves fingerprint quality from packed `NFRecord`.

```
public static byte GetQuality(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

The value of fingerprint quality.

7.8.1.1.32. GetSize Methods

Calculates packed size of `NFRecord`.

7.8.1.1.32.1. GetSize()

Calculates packed size of NFRecord.

```
public int GetSize();
```

Return Values

The packed size of NFRecord.

Remarks

The method calculates current (2.0) version packed size of NFRecord.

See Also

NFRecord Constructors | `GetSize(uint flags)`

7.8.1.1.32.2. GetSize(uint flags)

Calculates packed size of NFRecord. Behavior is controlled through flags.

```
public int GetSize(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The packed size of NFRecord.

Remarks

The method calculates current (2.0) version packed size of NFRecord.

See Also

NFRecord Constructors | `GetSize()`

7.8.1.1.33. GetSizeV1 Method

Calculates version 1.0 packed size of NFRecord.

7.8.1.1.33.1. GetSizeV1()

Calculates version 1.0 packed size of NFRecord.

```
public int GetSizeV1()
```

Return Values

The packed size of NFRecord.

7.8.1.1.33.2. GetSizeV1(uint)

Calculates version 1.0 packed size of NFRecord. Behavior is controlled through flags.

```
public int GetSizeV1(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The packed size of NFRecord.

See Also

NFRecord Constructors | `GetSize`

7.8.1.1.34. GetVertResolution Method

Retrieves vertical resolution of fingerprint image from packed `NFRecord`.

```
public static ushort GetVertResolution(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

The vertical resolution of fingerprint image.

See Also

GetHorzResolution

7.8.1.1.35. GetWidth Method

Retrieves width of fingerprint image from packed `NFRecord`.

```
public static ushort GetWidth(
        byte[] buffer
);
```

Parameters

buffer	The byte array of packed `NFRecord`.

Return Values

The width of fingerprint image.

See Also

GetHeight

7.8.1.1.36. Save Methods

Packs `NFRecord` to byte array.

7.8.1.1.36.1. Save()

Packs NFRecord to byte array.

```
public byte[] Save();
```

Return Values

The array of packed NFRecord.

Remarks

The method packs NFRecord in current (2.0) version format.

Note that blocked orientations are not packed. To pack them use Save with flags or SaveV1

method.

See Also

NFRecord Constructors

7.8.1.1.36.2. Save(uint flags)

Packs NFRecord to byte array. Behavior is controlled through flags.

```
public byte[] Save(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The byte array of packed NFRecord.

Remarks

The method packs NFRecord in current (2.0) version format.

Note that blocked orientations are not packed by default.

The following flags are supported:

See Also

NFRecord Constructors | Save | SaveV1

7.8.1.1.37. SaveV1 Method

Packs `NFRecord` to memory buffer in version 1.0 format.

7.8.1.1.37.1. SaveV1()

```
public byte[] SaveV1();
```

Return Values

The byte array of packed NFRecord.

See Also

Save

7.8.1.1.37.2. SaveV1(uint)

Packs NFRecord to memory buffer in version 1.0 format.

```
public byte[] SaveV1(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The byte array of packed NFRecord.

Remarks

Note that blocked orientations are not packed by default.

The following flags are supported: FlagSkipSingularPoints, FlagSaveBlockedOrients, FlagSkipCurvatures, FlagSkipGs.

See Also

NFRecord Constructors | Save

7.8.1.2. NFRecord.CoreCollection Class

Represents the collection of NFCore.

```
public sealed class CoreCollection: IList
```

Properties

Capacity	Gets or sets the number of elements that the CoreCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Copyright © 2006 Neurotechnologija

Methods

Add	Adds an object to the end of the CoreCollection.
Clear	Removes all elements from the CoreCollection.
`CopyTo`	Copies the CoreCollection or a portion of it to a onedimensional array.
`GetEnumerator`	Returns an enumerator that can be used to iterate through the CoreCollection.
Insert	Inserts an element into the CoreCollection at the specified index.
RemoveAt	Removes the element at the specified index within the `NFRecord.CoreCollection`.

7.8.1.2.1. Capacity Property

Gets or sets the number of elements that the CoreCollection can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the `NFRecord.CoreCollection` can contain.

See Also

Count

7.8.1.2.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection.

7.8.1.2.3. NFRecord.CoreCollection.Item Property

Gets or sets the member from collection by index.

```
public NFCore this[
        int index
] {get; set;}
```

Parameters

index	The index of the NFCore structure to retrieve from the collection.

Property value

A NFCore structure.

See also

NFCore Count

7.8.1.2.4. Add Method

Adds an object to the end of the CoreCollection.

```
public int Add(
        NFCore value
);
```

Parameters

value	The NFCore to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

7.8.1.2.5. Clear Method

Removes all elements from the CoreCollection.

```
public void Clear();
```

See Also

RemoveAt

7.8.1.2.6. CopyTo Method

Copies the CoreCollection or a portion of it to a onedimensional array.

```
public void CopyTo(Array array, int index)
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

7.8.1.2.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the CoreCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the checked index collection.

See Also

NFRecord.CoreCollection

7.8.1.2.8. Insert Method

Inserts an element into the CoreCollection at the specified index.

```
public void Insert(
        int index,
        NFCore value
);
```

Parameters

index	The zero-based index location where the NFCore is inserted.

Copyright © 2006 Neurotechnologija

value	The NFCore to add to the collection.

See Also

Add

7.8.1.2.9. RemoveAt Method

Removes the element at the specified index within the `NFRecord.CoreCollection`.

```
public void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the NFCore to remove.

See Also

Clear

7.8.1.3. NFRecord.DeltaCollection Class

Represents the collection of NFDelta.

```
public sealed class DeltaCollection: IList
```

Properties

Capacity	Gets or sets the number of elements that the FmrFingerView.MinutiaCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the `NFRecord.DeltaCollection`.

Copyright © 2006 Neurotechnologija

Clear	Removes all elements from the NFRecord.DeltaCollection.
CopyTo	Copies the DeltaCollection or a portion of it to a onedimensional array.
GetEnumerator	Returns an enumerator that can be used to iterate through the DeltaCollection.
Insert	Inserts an element into the NFRecord.DeltaCollection at the specified index.
RemoveAt	Removes the element at the specified index within the NFRecord.DeltaCollection.

7.8.1.3.1. Capacity Property

Gets or sets the number of elements that the FmrFingerView.MinutiaCollection can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the NFRecord.DeltaCollection can contain.

See Also

Count

7.8.1.3.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection.

7.8.1.3.3. NFRecord.DeltaCollection.Item Property

Gets or sets the member from collection by index.

```
public NFDelta this[
        int index
] {get; set;}
```

Copyright © 2006 Neurotechnologija

Parameters

index	The index of the NFDelta structure to retrieve from the collection.

Property value

A NFDelta structure.

See also

Count

7.8.1.3.4. Add Method

Adds an object to the `NFRecord.DeltaCollection`.

```
public int Add(
        NFDelta value
);
```

Parameters

value	The NFDelta to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

7.8.1.3.5. Clear Method

Removes all elements from the `NFRecord.DeltaCollection`.

```
public virtual void Clear();
```

See Also

RemoveAt

7.8.1.3.6. CopyTo Method

Copyright © 2006 Neurotechnologija

Copies the DeltaCollection or a portion of it to a onedimensional array.

```
public void CopyTo(Array array, int index)
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

7.8.1.3.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the DeltaCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the core collection.

7.8.1.3.8. Insert Method

Inserts an element into the `NFRecord.DeltaCollection` at the specified index.

```
public void Insert(
        int index,
        NFDelta value
);
```

Parameters

index	The zero-based index location where the NFDelta is inserted.
value	The NFDelta to add to the collection.

See Also

Add

7.8.1.3.9. RemoveAt Method

Removes the element at the specified index within the `NFRecord.DeltaCollection`.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the NFDelta to remove.

See Also

Clear

7.8.1.4. NFRecord.DoubleCoreCollection Class

Represents the collection of NFDoubleCore.

```
public sealed class DoubleCoreCollection: IList
```

Properties

Capacity	Gets or sets the number of elements that the NFRecord.DoubleCoreCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the `NFRecord.DoubleCoreCollection`.
Clear	Removes all elements from the `NFRecord.DoubleCoreCollection`.
`CopyTo`	Copies the DoubleCoreCollection or a portion of it to a onedimensional array.
`GetEnumerator`	Returns an enumerator that can be used to iterate through the DoubleCoreCollection.

Copyright © 2006 Neurotechnologija

Insert	Inserts an element into the `NFRecord.DoubleCoreCollection` at the specified index.
RemoveAt	Removes the element at the specified index of the `NFRecord.DoubleCoreCollection`.

7.8.1.4.1. Capacity Property

Gets or sets the number of elements that the NFRecord.DoubleCoreCollection can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the `NFRecord.DoubleCoreCollection` can contain.

See Also

Count

7.8.1.4.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection.

7.8.1.4.3. Item Property

Gets or sets the member from collection by index.

```
public NFDoubleCore this[
        int index
        ] {get; set;}
```

Property value

A NFDoubleCore structure.

See also

NFDoubleCore Count

7.8.1.4.4. Add Method

Adds an object to the `NFRecord.DoubleCoreCollection`.

```
public int Add(
        NFDoubleCore value
);
```

Parameters

value	The NFDoubleCore to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

7.8.1.4.5. Clear Method

Removes all elements from the `NFRecord.DoubleCoreCollection`.

```
public virtual void Clear();
```

See Also

RemoveAt

7.8.1.4.6. CopyTo Method

Copies the DoubleCoreCollection or a portion of it to a onedimensional array.

```
public void CopyTo(Array array, int index)
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

7.8.1.4.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the DoubleCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the double core collection.

7.8.1.4.8. Insert Method

Inserts an element into the `NFRecord.DoubleCoreCollection` at the specified index.

```
public void Insert(
        int index,
        NFDoubleCore value
);
```

Parameters

index	The zero-based index location where the NFDoubleCore is inserted.
value	The NFDoubleCore to add to the collection.

See Also

Add

7.8.1.4.9. RemoveAt Method

Removes the element at the specified index of the `NFRecord.DoubleCoreCollection`.

```
public virtual void RemoveAt(int index);
```

Parameters

index	The zero-based index of the NFDoubleCore to remove.

See Also

Clear

7.8.1.5. NFRecord.MinutiaCollection Class

Represents the collection of NFMinutia.

```
public sealed class MinutiaCollection: IList
```

Properties

Capacity	Gets or sets the number of elements that the NFRecord.MinutiaCollection can contain.
Count	Gets the number of items in the collection.
Item	Gets or sets the member from collection by index.

Methods

Add	Adds an object to the `NFRecord.MinutiaCollection`.
Clear	Removes all elements from the `NFRecord.MinutiaCollection`.
`CopyTo`	Copies the MinutiaCollection or a portion of it to a one- dimensional array.
`GetEnumerator`	Returns an enumerator that can be used to iterate through the MinutiaCollection.
Insert	Inserts an element into the `NFRecord.MinutiaCollection` at the specified index.
RemoveAt	Removes the element at the specified index within the `NFRecord.MinutiaCollection`.

7.8.1.5.1. Capacity Property

Gets or sets the number of elements that the NFRecord.MinutiaCollection can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the `NFRecord.MinutiaCollection` can contain.

See Also

Count

7.8.1.5.2. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection.

7.8.1.5.3. MinutiaCollection.Item Property

Gets or sets the member from collection by index.

```
public NFMinutia this[
        int index
] {get; set;}
```

Parameters

index	The index of the NFMinutia structure to retrieve from the collection.

Property value

A NFMinutia structure.

See also

NFMinutia | Count

7.8.1.5.4. Add Method

Adds an object to the `NFRecord.MinutiaCollection`.

```
public int Add(
        NFMinutia value
);
```

Parameters

value	The NFMinutia to add to the collection.

Return Values

The zero-based index into the collection where the item was added.

See Also

Insert

7.8.1.5.5. Clear Method

Removes all elements from the `NFRecord.MinutiaCollection`.

```
public virtual void Clear()
```

See Also

RemoveAt

7.8.1.5.6. CopyTo Method

Copies the MinutiaCollection or a portion of it to a one- dimensional array.

```
public void CopyTo(Array array, int index)
```

Parameters

array	The one-dimensional array that is the destination of the elements copied from this collection.
index	The zero-based index in array at which copying begins.

7.8.1.5.7. GetEnumerator Method

Returns an enumerator that can be used to iterate through the MinutiaCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the minutia collection.

7.8.1.5.8. Insert Method

Inserts an element into the `NFRecord.MinutiaCollection` at the specified index.

```
public void Insert(
        int index,
        NFMinutia value
);
```

Parameters

index	The zero-based index location where the NFMinutia is inserted.
value	The NFMinutia to add to the collection.

See Also

Add

7.8.1.5.9. RemoveAt Method

Removes the element at the specified index within the `NFRecord.MinutiaCollection`.

```
public virtual void RemoveAt(
        int index
);
```

Parameters

index	The zero-based index of the NFMinutia to remove.

See Also

Clear

7.8.1.6. NFRecord.MinutiaNeighboursCollection Class

Represents the collection of NFMinutiaNeighbour.

Properties

Count	Gets or sets the number of elements that the

	MinutiaNeighboursCollection can contain.
Item	Gets the member of items in the collection.

Methods

GetCount	Retrieves the number of minutia neighbours at the specified index.
GetEnumerator	Returns an enumerator that can be used to iterate through the MinutiaNeighboursCollection.

7.8.1.6.1. Count Property

Gets or sets the number of elements that the MinutiaNeighboursCollection can contain.

```
public int Count {get;}
```

Property value

The number of items in the collection

7.8.1.6.2. NFRecord.MinutiaNeighboursCollection.Item Property

7.8.1.6.2.1. this[int]

Gets the members array from collection by index;

```
public NFMinutiaNeighbour[] this[
                int minutiaIndex
        ] {get;}
```

Parameters

minutiaIndex	The index of the specified minutia.

Property value

A NFMinutiaNeighbour structures array of specified minutia.

See Also

NFMinutiaNeighbour | Count

7.8.1.6.2.2. this[int, int]

Gets or sets the member from collection by index

```
public NFMinutiaNeighbour this[
                int minutiaIndex,
                int index
        ] {get; set;}
```

Parameters

minutiaIndex	The index of the specified minutia.
index	The index of the specified neighbour minutia to retrieve from the collection.

Property value

A NFMinutiaNeighbour structure of specified neighbour minutia.

See Also

NFMinutiaNeighbour | >Count

7.8.1.6.3. GetCount Method

Retrieves the number of minutia neighbours at the specified index.

```
public int GetCount(
        int minutiaIndex
);
```

Parameters

minutiaIndex	The index of the minutia.

Return Values

The number of the minutia neighbours.

7.8.1.6.4. GetEnumerator Method

Returns an enumerator that can be used to iterate through the MinutiaNeighboursCollection.

```
public IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the minutia neighbours collection.

7.8.1.7. NFCore Struct

The structure contains information of core.

```
public struct NFCore
```

Constructors

NFCore Constructor	Initializes a new instance of the `NFCore` structure.

Properties

Angle	Gets or sets Angle of core.
RawAngle	Gets or sets raw angle of core
X	Gets or sets y coordinate of core.
Y	Gets or sets x coordinate of core.

7.8.1.7.1. NFCore Constructor

7.8.1.7.1.1. NFCore(ushort, ushort)

Initializes a new instance of the NFCore structure.

```
public NFCore(
        ushort x,
        ushort y
);
```

Parameters

ushort	The x - coordinate of the core.
ushort	The y - coordinate of the core.

7.8.1.7.1.2. NFCore(ushort x, ushort y, int angle)

```
public NFCore(
        ushort x,
        ushort y,
        int angle
);;
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.
angle	The angle of the core.

7.8.1.7.1.3. NFCore(ushort x, ushort y, double angle)

```
public NFCore(
        ushort x,
        ushort y,
        double angle
);
```

Parameters

x	The x - coordinate of the core.
y	The y - coordinate of the core.
angle	The angle of the core.

7.8.1.7.2. Angle Property

Gets or sets Angle of core.

```
public double Angle {get; set;}
```

Property value

The angle of the core.

7.8.1.7.3. RawAngle Property

Gets or sets raw angle of core

```
public int RawAngle {get; set}
```

Property value

The raw angle of the core.

Remarks

The angle of the core is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the angle of the core is unknown.

7.8.1.7.4. X Property

Gets or sets y coordinate of core.

```
public int X {get; set;}
```

Property value

The y coordinate of the core.

Remarks

The x coordinate of the core is specified in pixels at Resolution and X * [NFRecord horizontal resolution] / `Resolution` can not be greater than MaxDimension or NFRecord width minus one.

7.8.1.7.5. Y Property

Gets or sets x coordinate of core.

```
public int Y {get; set;}
```

Property value

The y coordinate of the core.

Remarks

The y coordinate of the core is specified in pixels at Resolution and Y * [NFRecord vertical resolution] / `Resolution` can not be greater than MaxDimension or NFRecord width minus one.

7.8.1.8. NFDelta Struct

The structure contains information of delta.

```
public struct NFDelta
```

Constructors

NFDelta Constructor	Initializes a new instance of the NFDelta structure.

Properties

Angle1	Gets or sets the first angle of delta.
Angle2	Gets or sets the second angle of delta.
Angle3	Gets or sets the third angle of delta.
RawAngle1	Gets or sets the first row angle of the delta.
RawAngle2	Gets or sets the second row angle of the delta.
RawAngle3	Gets or sets the third row angle of the delta.
X	Gets or sets x coordinate of delta.
Y	Gets or sets y coordinate of delta.

7.8.1.8.1. NFDelta Constructor

Initializes a new instance of the NFDelta structure.

7.8.1.8.1.1. NFDelta(ushort x, ushort y)

```
public NFDelta(
        ushort x,
        ushort y
);
```

Parameters

x	The x - coordinate of the delta.
y	The y - coordinate of the delta.

7.8.1.8.1.2. NFDelta(ushort x, ushort y, int angle1, int angle2, int angle3)

```
public NFDelta(
```

```
        ushort x,
        ushort y,
        int angle1,
        int angle2,
        int angle3
);
```

Parameters

x	The x - coordinate of the delta.
y	The y - coordinate of the delta.
angle1	The first angle of the delta.
angle2	The second angle of the delta.
angle3	The third angle of the delta.

7.8.1.8.1.3. NFDelta(ushort x, ushort y, double angle1, double angle2, double angle3)

```
public NFDelta(
        ushort x,
        ushort y,
        double angle1,
        double angle2,
        double angle3
);
```

Parameters

x	The x - coordinate of the delta.
y	The y - coordinate of the delta.
angle1	The first angle of the delta.
angle2	The second angle of the delta.
angle3	The third angle of the delta.

7.8.1.8.2. Angle1 Property

Gets or sets the first angle of delta.

```
public double Angle1 {get; set;}
```

Property value

The first angle of the delta.

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

7.8.1.8.3. Angle2 Property

Gets or sets the second angle of delta.

```
public double Angle2 {get; set;}
```

Property value

The second angle of the delta.

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

7.8.1.8.4. Angle3 Property

Gets or sets the third angle of delta.

```
public double Angle3 {get; set;}
```

Property value

The third angle of the delta.

Remarks

The angle of the delta is specified in 180/128 degrees units in counterclockwise order and can not be less than zero or greater than 256 minus one.

The value of -1 can be specified if the first angle of the delta is unknown.

7.8.1.8.5. RawAngle1 Property

```
public int RawAngle1 {get; set;}
```

Property value

The first raw angle of the delta.

7.8.1.8.6. RawAngle2 Property

```
public int RawAngle2 {get; set;}
```

Property value

The second raw angle of the delta.

7.8.1.8.7. RawAngle3 Property

```
public int RawAngle3 {get; set;}
```

Property value

The third raw angle fog the delta.

7.8.1.8.8. X Property

Gets or sets x coordinate of delta.

```
public int X {get; set;}
```

Property value

The x coordinate of the delta.

Remarks

The x coordinate of the delta is specified in pixels at Resolution and X * [NFRecord horizontal resolution] / `Resolution` can not be greater than MaxDimension or NFRecord width minus one.

7.8.1.8.9. Y Property

Gets or sets y coordinate of delta.

```
public int Y {get; set;}
```

Property value

The x coordinate of the delta.

Remarks

The y coordinate of the delta is specified in pixels at Resolution and Y * [NFRecord vertical resolution] / `Resolution` can not be greater than MaxDimension or NFRecord width minus one.

7.8.1.9. NFDoubleCore Struct

The structure contains information of double core.

```
public struct NFDoubleCore
```

Constructors

NFDoubleCore Constructor	Initializes a new instance of the `NF-DoubleCore` structure.

Properties

X	Gets or sets x coordinate of the double core.
Y	Gets or sets y coordinate of the double core.

7.8.1.9.1. NFDoubleCore Constructor

```
public NFDoubleCore(ushort x, ushort y);
```

Parameters

x	The x - coordinate of the double core.
y	The y - coordinate of the double core.

7.8.1.9.2. X Property

Gets or sets x coordinate of the double core.

```
public int X {get; set;}
```

Property value

The X coordinate of the double core.

7.8.1.9.3. Y Property

Gets or sets y coordinate of the double core.

```
public int Y {get; set;}
```

Property value

The Y coordinate of the double core.

7.8.1.10. NFMinutia Struct

The structure contains information of minutia.

```
public struct NFMinutia
```

Constructors

NFMinutia Constructor	Initializes a new instance of the NFMinutia structure.

Properties

Angle	Gets or sets the angle of the minutia.
Curvature	Gets or sets the ridge curvature near minutia.
G	Gets or sets the G (ridge density) near minutia.
Quality	Gets or sets quality of the minutia.
RawAngle	Gets or sets the raw angle of the minutia.
Type	Gets or sets the type of the minutia.
X	Gets or sets x coordinate of the minutia.
Y	Gets or sets y coordinate of the minutia.

7.8.1.10.1. NFMinutia Constructor

Initializes a new instance of the NFMinutia structure.

7.8.1.10.1.1. NFMinutia(ushort x, ushort y, NFMinutiaType type, byte angle)

```
public NFMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        byte angle
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

7.8.1.10.1.2. NFMinutia(ushort x, ushort y, NFMinutiaType type, double angle)

```
public NFMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        double angle
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.

7.8.1.10.1.3. NFMinutia(ushort x, ushort y, NFMinutiaType type, byte angle, byte quality, byte curvature, byte g)

```
public NFMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        byte angle,
        byte quality,
        byte curvature,
```

```
        byte g
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.
quality	The quality of the minutia.
curvature	The ridge curvature near minutia.
g	The G (ridge density) near minutia.

7.8.1.10.1.4. NFMinutia(ushort x, ushort y, NFMinutiaType type, double angle, byte quality, byte curvature, byte g)

```
public NFMinutia(
        ushort x,
        ushort y,
        NFMinutiaType type,
        double angle,
        byte quality,
        byte curvature,
        byte g
);
```

Parameters

x	The x - coordinate of the minutia.
y	The y - coordinate of the minutia.
type	One of the NFMinutiaType values.
angle	The angle of the minutia.
quality	The quality of the minutia.
curvature	The ridge curvature near minutia.
g	The G (ridge density) near minutia.

Copyright © 2006 Neurotechnologija

7.8.1.10.2. Angle Property

Gets or sets the angle of the minutia.

```
public double Angle {get; set;}
```

Property value

The angle of the minutia.

Remarks

The angle of the minutia is specified in 180/128 degrees units in counterclockwise order and can not be greater than 256 minus one.

7.8.1.10.3. Curvature Property

Gets or sets the ridge curvature near minutia.

```
public byte Curvature {get; set;}
```

Property value

The ridge curvature near minutia.

Remarks

If curvature of the minutia is unknown it must be set to 255.

7.8.1.10.4. G Property

Gets or sets the G (ridge density) near minutia.

```
public byte G {get; set;}
```

Property value

The G (ridge density) near minutia.

Remarks

If G of the minutia is unknown it must be set to 255.

7.8.1.10.5. Quality Property

Gets or sets quality of the minutia.

```
public byte Quality {get; set;}
```

Property value

The quality of the minutia.

Remarks

The quality of the minutia must be in the range [0, 100]. The higher it is, the better the quality of the minutia is.

If quality of the minutia is unknown it must be set to zero.

7.8.1.10.6. RawAngle Property

Gets or sets the raw angle of the minutia.

```
public byte RawAngle {get; set;}
```

Property value

The raw angle of the minutia.

7.8.1.10.7. Type Property

Gets or sets the type of the minutia.

```
public NFMinutiaType Type {get; set;}
```

Property value

One of the NFMinutiaType values.

7.8.1.10.8. X Property

Gets or sets x coordinate of the minutia.

```
public int X {get; set;}
```

Property value

The X coordinate of the minutia.

Remarks

The x coordinate of the minutia is specified in pixels at `Resolution` and *X* * [NFRecord horizontal resolution] / `Resolution` can not be greater than `MaxDimension` or NFRecord width minus one.

7.8.1.10.9. Y Property

Copyright © 2006 Neurotechnologija

Gets or sets y coordinate of the minutia.

```
public int Y {get; set;}
```

Property value

The Y coordinate of the minutia.

Remarks

The y coordinate of the minutia is specified in pixels at `Resolution` and *Y* * [NFRecord vertical resolution] / `Resolution` can not be greater than `MaxDimension` or NFRecord width minus one.

7.8.1.11. NFMinutiaNeighbour Struct

The structure contains information of minutia neighbour.

```
public struct NFMinutiaNeighbour
```

Constructors

NFMinutiaNeighbour Constructor	Initializes a new instance of the `NFMinutiaNeighbour` structure.

Fields

Empty	Represents a minutia neighbour that is null reference.

Properties

Index	Gets or sets the index of minutia neighbour.
RidgeCount	Gets or sets the ridge count between the minutia and minutia neighbour.

7.8.1.11.1. NFMinutiaNeighbour Constructor

```
public NFMinutiaNeighbour(
        int index,
        byte ridgeCount
);
```

Parameters

index	The index of neighbour.
ridgeCount	The ridge count of neighbour.

7.8.1.11.2. Empty Field

Represents a NFMinutiaNeighbour that is a null reference.

```
public static readonly NFMinutiaNeighbour Empty;
```

7.8.1.11.3. Index Property

Gets or sets the index of minutia neighbour.

```
public int Index {get; set;}
```

Property value

The index of minutia neighbour.

7.8.1.11.4. RidgeCount Property

Gets or sets the ridge count between the minutia and minutia neighbour.

```
public int RidgeCount {get; set;}
```

Property value

The ridge count between the minutia and minutia neighbour.

7.8.1.12. NFImpressionType Enumeration

Specifies the impression types.

```
public enum NFImpressionType
```

Members

Member name	Description
LatentImpression	Latent impression fingerprint.
LatentLift	Latent lift fingerprint.

Member name	Description
LatentPhoto	Latent photo fingerprint.
LatentTracing	Latent tracing fingerprint.
LiveScanContactless	Live-scanned fingerprint using contactless device.
LiveScanPlain	Live-scanned plain fingerprint.
LiveScanRolled	Live-scanned rolled fingerprint.
NonliveScanPlain	Nonlive-scanned (from paper) plain fingerprint.
NonliveScanRolled	Nonlive-scanned (from paper) rolled fingerprint.
Swipe	Live-scanned fingerprint by sliding the finger across a "swipe" sensor.

7.8.1.13. NFMinutiaFormat Enumeration

Specifies formats of minutia. This enumeration allows a bitwise combination of its member values.

```
public enum NFMinutiaFormat
```

Members

Member name	Description
HasCurvature	If near minutia is ridge curvature
HasG	If near minutia is G(ridge density).
HasQuality	The quality of the minutia.

7.8.1.14. NFMinutiaType Enumeration

Specifies types of minutia.

```
public enum NFMinutiaType
```

Members

Member name	Description
Bifurcation	The minutia that is a bifurcation of a ridge.
End	The minutia that is an end of a ridge.
Unknown	The type of the minutia is unknown.

7.8.1.15. NFPatternClass Enumeration

Specifies pattern class of the fingerprint.

```
public enum NFPatternClass
```

Members

Member name	Description
AccidentalWhorl	Accidental whorl pattern class.
Amputation	Amputation. Pattern class is not available.
CentralPocketLoop	Central pocket loop pattern class.
DoubleLoop	Double loop pattern class.
LeftSlantLoop	Left slant loop pattern class.
PlainArch	Plain arch pattern class.
PlainWhorl	Plain whorl pattern class.
RadialLoop	Radial loop pattern class.
RightSlantLoop	Right slant loop pattern class.
Scar	Scar. Pattern class is not available.
TentedArch	Tented arch pattern class.
UlnarLoop	Ulnar loop pattern class.
Unknown	Unknown pattern class.
Whorl	Whorl pattern class.

Remarks

This enumeration is implemented according to ANSI/NIST-ITL 1-2000 standard.

7.8.1.16. NFPosition Enumeration

Specifies finger position.

```
public enum NFPosition
```

Members

Member name	Description
LeftIndex	Index finger of the left hand.
LeftLittle	Little finger of the left hand.
LeftMiddle	Middle finger of the left hand.
LeftRing	Ring finger of the left hand.
LeftThumb	Thumb of the left hand.
RightThumb	Thumb of the right hand.
RightIndex	Index finger of the right hand.
RightLittle	Little finger of the right hand.
RightMiddle	Middle finger of the right hand.
RightRing	Ring finger of the right hand.
Unknown	Unknown finger.

7.8.1.17. NFRidgeCountsType Enumeration

Specifies type of ridge counts contained in NFRecord.

```
public enum NFRidgeCountsType
```

Members

Member name	Description
EightNeighbours	The NFRecord contains ridge counts to closest minutia in each of the eight sectors of each minutia. First sector starts at minutia angle.

Member name	Description
EightNeighboursWithIndexes	The NFRecord contains ridge counts to eight neighbours of each minutia.
FourNeighbours	The NFRecord contains ridge counts to closest minutia in each of the four sectors of each minutia. First sector starts at minutia angle.
FourNeighboursWithIndexes	The NFRecord contains ridge counts to four neighbours of each minutia.
None	The NFRecord does not contain ridge counts.
Unspecified	For internal use.

7.9. Neurotec.Biometrics.NfsMatcher Library

Provides functionality for comparing Neurotechnologija Fingers Templates.

DLL: `Neurotec.Biometrics.NfsMatcher.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for comparing Neurotechnologija Fingers Templates.

7.9.1. Neurotec.Biometrics Namespace

Provides functionality for comparing Neurotechnologija Fingers Templates.

Classes

NfsmFingerMatchDetails	Specifies the fingers matching information.
NfsmMatchDetails	Specifies the matching information.
NfsmMatchDetails.NfsmFingerMatchDetailsCollection	Represents the collection of NfsmFingerMatchDetails.
NfsMatcher	Provides methods and properties for comparing Neurotechnologija Fingers Templates (NFTemplates), encapsulated in Neurotechnologija Fingers Matcher (NfsMatcher) ob-

	ject.

Enumerations

NfsmUsedMatcher	Specifies the used algorithm.

7.9.1.1. NfsmFingerMatchDetails Class

Specifies the fingers matching information.

Properties

MatchedIndex	Specifies the zero-based index in the second NFTemplate.
NfmMatchDetails	Specifies structure NfmMatchDetails that receives finger match details.
Score	Specifies score of matched fingerprints.
VfmMatchDetails	Specifies structure VfmMatchDetails that receives finger match details.

Methods

Dispose	Releases the resources used by NfsmFinger-MatchDetails.

7.9.1.1.1. MatchedIndex Property

Specifies the zero-based index in the second NFTemplate.

```
public int MatchedIndex {get;}
```

Property value

The index of fingerprint template in NFTemplate.

Remarks

-1 no match found. The second template in the identification comes from the IdentifyNext method. The first template comes from the IdentifyStart method.

See Also

NFTemplate

7.9.1.1.2. NfmMatchDetails Property

Specifies structure NfmMatchDetails that receives finger match details.

```
public NfmMatchDetails NfmMatchDetails {get;}
```

Property value

The NfmMatchDetails object.

See Also

NFTemplate | NfmMatchDetails

7.9.1.1.3. Score Property

Specifies score of matched fingerprints.

```
public int Score {get;}
```

Property value

The score of matched NFTemplates.

See Also

NFTemplate

7.9.1.1.4. VfmMatchDetails Property

Specifies structure VfmMatchDetails that receives finger match details.

```
public VfmMatchDetails VfmMatchDetails {get;}
```

Property value

The VfmMatchDetails object.

See Also

NFTemplate | VfmMatchDetails

7.9.1.1.5. Dispose Method

Copyright © 2006 Neurotechnologija

Releases the resources used by NfsmFingerMatchDetails.

```
public void Dispose();
```

7.9.1.2. NfsmMatchDetails Class

Specifies the matching information.

Properties

Fingers	Gets the collecton of NfsmFingerMatchDetails structures.
Score	Gets the score of the matched fingerprints.

7.9.1.2.1. Fingers Property

Gets the collecton of NfsmFingerMatchDetails structures.

```
public NfsmMatchDetails.NfsmFingerMatchDetailsCollection Fingers {get;}
```

Property value

The NfsmMatchDetails.NfsmFingerMatchDetailsCollection object.

See Also

NfsmMatchDetails.NfsmFingerMatchDetailsCollection

7.9.1.2.2. Score Property

Gets the score of the matched fingerprints.

```
public int Score {get;}
```

Property value

The score of the matched fingerprints.

7.9.1.3. NfsmMatchDetails.NfsmFingerMatchDetailsCollection Class

Represents the collection of NfsmFingerMatchDetails.

Properties

Count	Gets the number of items in the collection.
Item	Gets the member from collection by index.

Methods

Contains	Determines whether the collection containts a NfsmFingerMatchDetails.
GetEnumerator	Returns an enumerator that can be used to iterate through the NfsmMatchDetails.NfsmFingerMatchDetailsCollection.
IndexOf	Returns the index within the collection of the finger match details.

7.9.1.3.1. Count Property

Gets the number of items in the collection.

```
public int Count {get;}
```

Property value

The number of items in the collection

7.9.1.3.2. NfsmMatchDetails.NfsmFingerMatchDetailsCollection.Item Property

Gets the member from collection by index.

```
public NfsmFingerMatchDetails this[
        int index
] {get;}
```

Parameters

index	The index of the NfsmFingerMatchDetails structure in the collection to get.

Property value

A NfsmFingerMatchDetails structure.

See Also

Count

7.9.1.3.3. Contains Method

Determines whether the collection containts a NfsmFingerMatchDetails.

```
public bool Contains(
        NfsmFingerMatchDetails value
);
```

Parameters

value	The NfsmFingerMatchDetails to locate in the `NfsmMatchDetails.NfsmFingerMatchDetailsCollection`

Return Values

`true` if the finger match details is found in the `NfsmMatchDetails.NfsmFingerMatchDetailsCollection`; otherwise, `false`.

See Also

NfsmFingerMatchDetails | IndexOf

7.9.1.3.4. GetEnumerator Method

```
public virtual IEnumerator GetEnumerator();
```

Return Values

An IEnumerator that represents the match details collection.

7.9.1.3.5. IndexOf Method

Returns the index within the collection of the finger match details.

```
public int IndexOf(
        NfsmFingerMatchDetails value
);
```

Parameters

value	A NfsmFingerMatchDetails object.

Return Values

The zero-based index of the NfsmFingerMatchDetails in the collection.

See Also

NfsmFingerMatchDetails

7.9.1.4. NfsMatcher Class

Provides methods and properties for comparing Neurotechnologija Fingers Templates (NFTemplates), encapsulated in Neurotechnologija Fingers Matcher (NfsMatcher) object.

Constructors

NfsMatcher Constructor	Initializes a new instance of the `NfsMatcher` class.

Properties

IsRegistered	Gets a value indicating whether Neurotec.Biometrics.NfsMatcher library registered.
MatchingThreshold	Gets or sets the matching threshold.
NfmMatchingSpeed	Gets or sets the NfmSpeed matching speed.
NfmMaximalRotation	Gets or sets maximal rotation.
NfmMode	Gets or sets scanners mode.
UsedFingerMatcher	Gets or sets one of the NfsmUsedMatcher finger matcher.
VfmMatchingSpeed	Gets or sets finger matcher speed.
VfmMaximalRotation	Gets or sets tha maximal rotation of fingerprint.
VfmMode	Gets or sets scanners mode.

Methods

Copyright © 2006 Neurotechnologija

CopyParameters	Copies parameters values from one NfsMatcher object to another NfsMatcher object.
Dispose	Releases the resources used by `NfsMatcher`.
GetParameter	Gets value of parameter by parameter id.
GetStaticParameter	Gets value of static parameter by parameter id.
IdentifyEnd	Ends identification process.
IdentifyNext	Compares the specified NFTemplate with one identification was started with.
IdentifyStart	Starts identification.
Register	Registers Neurotec.Biometrics.NfsMatcher library.
Reset	Resets all NfsMatcher parameters to default values.
SetParameter	Sets value of parameter by parameter id.
SetStaticParameter	Sets value of static parameter by parameter id.
Unregister	Unregisters Neurotec.Biometrics.NfsMatcher library.
Verify	Compares two NFTempalte.

Constants

DllName	Name of DLL containing unmanaged part of this class.
ParameterCopyright	Identifier specifying library copyright static read-only parameter of type string.
ParameterMatchingThreshold	Identifier specifying matching threshold (biggest allowed FAR) parameter of type int. Parameter value is equal to - 12 * log10(FAR) and must be not less than zero (for example, 48 for FAR = 0.01%).
ParameterName	Identifier specifying library name static

Copyright © 2006 Neurotechnologija

	read-only parameter of type string.
ParameterUsedMatcher	Specifies used matcher. One of the NfsmUsedMatcher values.
ParameterVersionHigh	Identifier specifying high part of library version static read-only parameter of type int. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
ParameterVersionLow	Identifier specifying low part of library version static read-only parameter of type int. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.
PartNfm	Identifier specifying that retrieving or setting parameter for NFMatcher contained in NfsMatcher.
PartNone	Identifier specifying that retrieving or setting parameter for NfsMatcher.
PartVfm	Identifier specifying that retrieving or setting parameter for VFMatcher contained in NfsMatcher.

7.9.1.4.1. NfsMatcher Constructor

Initializes a new instance of the `NfsMatcher` class.

```
public NfsMatcher();
```

7.9.1.4.2. IsRegistered Property

Gets a value indicating whether Neurotec.Biometrics.NfsMatcher library registered.

```
public static bool IsRegistered {get;}
```

Property value

`true` if Neurotec.Biometrics.NfsMatcher library is registered; otherwise, `false`.

See Also

Register | Unregister

7.9.1.4.3. MatchingThreshold Property

Gets or sets the matching threshold.

```
public int MatchingThreshold {get; set;}
```

Property value

A matching threshold.

See Also

SetParameter | GetParameter

7.9.1.4.4. NfmMatchingSpeed Property

Gets or sets the NfmSpeed matching speed.

```
public NfmSpeed NfmMatchingSpeed {get; set;}
```

Property value

One of the NfmSpeed values. The default is `NfmSpeed.Low`.

See Also

NfmSpeed

7.9.1.4.5. NfmMaximalRotation Property

Gets or sets maximal rotation.

```
public byte NfmMaximalRotation {get; set;}
```

Property value

The maximal rotation.

7.9.1.4.6. NfmMode Property

Gets or sets scanners mode.

```
public uint NfmMode {get; set;}
```

Property value

One of the Mode constants.

Copyright © 2006 Neurotechnologija

See Also

Mode

7.9.1.4.7. UsedFingerMatcher Property

Gets or sets one of the NfsmUsedMatcher finger matcher.

```
public NfsmUsedMatcher UsedFingerMatcher {get; set;}
```

Property value

One of the NfsmUsedMatcher values.

See Also

NfsmUsedMatcher

7.9.1.4.8. VfmMatchingSpeed Property

Gets or sets finger matcher speed.

```
public VfmSpeed VfmMatchingSpeed {get; set;}
```

Property value

One of the VfmSpeed values. The default is `VfmSpeed.Low`.

See Also

VfmSpeed

7.9.1.4.9. VfmMaximalRotation Property

Gets or sets tha maximal rotation of fingerprint.

```
public byte VfmMaximalRotation {get; set;}
```

Property value

The maximal rotation of fingerprint.

7.9.1.4.10. VfmMode Property

Gets or sets scanners mode.

```
public uint VfmMode {get; set;}
```

Property value

One of the Mode constants.

See Also

Mode

7.9.1.4.11. CopyParameters Method

Copies parameters values from one NfsMatcher object to another NfsMatcher object.

```
public static void CopyParameters(
        NfsMatcher sourceMatcher,
        NfsMatcher destinationMatcher
);
```

Parameters

sourceMatcher	The NfsMatcher object.
destinationMatcher	The NfsMatcher object.

See Also

SetParameter | GetParameter

7.9.1.4.12. Dispose Method

Releases the resources used by `NfsMatcher`.

```
public void Dispose();
```

7.9.1.4.13. GetParameter Method

Gets value of parameter by parameter id.

```
public object GetParameter(
        ushort partId,
        ushort parameterId
);
```

Parameters

partId	One of the constants: `PartNfm`, `PartNone`, `PartVfm`.

parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

partId can be one of the following:

- PartNfm
- PartNone
- PartVfm

If *partId* is PartNfm, *parameterId* must be any NFMatcher GetParameter method supports except for ParameterMatchingThreshold.

If *partId* is PartVfm, *parameterId* must be any VFMatcher GetParameter method supports except for ParameterMatchingThreshold.

If *partId* is PartNone, the following values can be used for *parameterId*:

- ParameterMatchingThreshold
- ParameterUsedMatcher

See Also

SetParameter | CopyParameters | constants

7.9.1.4.14. GetStaticParameter Method

Gets value of static parameter by parameter id.

```
public static object GetStaticParameter(
        ushort partId,
        ushort parameterId
);
```

Parameters

partId	One of the constants: `PartNfm`, `PartNone`, `PartVfm`.
parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

partId can be one of the following:

- PartNfm
- PartNone
- PartVfm

If *partId* is PartNfm, *parameterId* must be any NFMatcher GetStaticParameter method supports.

If *partId* is PartVfm, *parameterId* must be any VFMatcher GetStaticParameter method supports.

If *partId* is PartNone, the following values can be used for *parameterId*:

- ParameterCopyright
- ParameterName
- ParameterVersionHigh
- ParameterVersionLow

See Also

SetStaticParameter | CopyParameters | constants

7.9.1.4.15. IdentifyEnd Method

Ends identification process.

```
public void IdentifyEnd();
```

See Also

IdentifyStart | IdentifyNext

7.9.1.4.16. IdentifyNext Method

Compares the specified NFTemplate with one identification was started with.

7.9.1.4.16.1. IdentifyNext(IntPtr,int)

```
public int IdentifyNext(
        IntPtr template,
```

```
        int templateSize
);
```

Parameters

template	Pointer to memory block that contains packed NFTemplate.
templateSize	The template size.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NFTemplate

7.9.1.4.16.2. IdentifyNext(IntPtr,int,NfsmMatchDetails)

```
public int IdentifyNext(
        IntPtr template,
        int templateSize,
        NfsmMatchDetails matchDetails
);
```

Parameters

template	Pointer to memory block that contains packed NFTemplate.
templateSize	The template size.
matchDetails	The NfsmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NfsmMatchDetails | NFTemplate

7.9.1.4.16.3. IdentifyNext(byte[])

```
public int IdentifyNext(
        byte[] template
);
```

Parameters

template	The byte array that contains packed NFTemplate.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NFTemplate

7.9.1.4.16.4. IdentifyNext(byte[],NfsmMatchDetails)

```
public int IdentifyNext(
        byte[] template,
        NfsmMatchDetails matchDetails
);
```

Parameters

template	The byte array that contains packed NFTemplate.

matchDetails	The NfsmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NfsmMatchDetails | NFTemplate

7.9.1.4.17. IdentifyStart Method

Starts identification.

7.9.1.4.17.1. IdentifyStart(IntPtr,int)

```
public void IdentifyStart(
        IntPtr template,
        int templateSize
);
```

Parameters

template	Pointer to memory block that contains packed NFTemplate.
templateSize	The template size.

See Also

IdentifyNext | IdentifyEnd | NFTemplate

7.9.1.4.17.2. IdentifyStart(IntPtr,int,out NfsmMatchDetails)

```
public void IdentifyStart(
        IntPtr template,
        int templateSize,
        out NfsmMatchDetails matchDetails
);
```

Parameters

template	Pointer to memory block that contains packed NFTemplate.
templateSize	The template size.
matchDetails	The NfsmMatchDetails object.

See Also

IdentifyNext | IdentifyEnd | NfsmMatchDetails | NFTemplate

7.9.1.4.17.3. IdentifyStart(byte[])

```
public void IdentifyStart(
        byte[] template
);
```

Parameters

template	The byte array that contains packed NFTemplate.

See Also

IdentifyNext | IdentifyEnd | NFTemplate

7.9.1.4.17.4. IdentifyStart(byte[],out NfsmMatchDetails)

```
public void IdentifyStart(
        byte[] template,
        out NfsmMatchDetails matchDetails
);
```

Parameters

template	The byte array that contains packed NFTemplate.
matchDetails	The NfsmMatchDetails object.

See Also

IdentifyNext | IdentifyEnd | NfsmMatchDetails | NFTemplate

7.9.1.4.18. Register Method

Registers Neurotec.Biometrics.NfsMatcher library.

```
public static void Register(
        string server
);
```

Parameters

server	The string that specifies license manager server name or IP address.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

7.9.1.4.19. Reset Method

Resets all NfsMatcher parameters to default values.

```
public void Reset();
```

7.9.1.4.20. SetParameter Method

Sets value of parameter by parameter id.

```
public void SetParameter(
        ushort partId,
        ushort parameterId,
        object value
);
```

Parameters

partId	One of the constants: `PartNfm`, `PartNone`, `PartVfm`.
parameterId	The parameter identifier.

value	The parameter value.

Return Values

The parameter value.

Remarks

partId can be one of the following:

- PartNfm
- PartNone
- PartVfm

If *partId* is PartNfm, *parameterId* must be any NFMatcher SetParameter method supports except for ParameterMatchingThreshold.

If *partId* is PartVfm, *parameterId* must be any VFMatcher SetParameter method supports except for ParameterMatchingThreshold.

If *partId* is PartNone, the following values can be used for *parameterId*:

- ParameterMatchingThreshold
- ParameterUsedMatcher

See Also

GetParameter | CopyParameters | constants

7.9.1.4.21. SetStaticParameter Method

Sets value of static parameter by parameter id.

```
public static void SetStaticParameter(
        ushort partId,
        ushort parameterId,
        object value
);
```

Parameters

partId	One of the constants: `PartNfm`, `PartNone`, `PartVfm`.
parameterId	The static parameter identifier.
value	The parameter value.

Copyright © 2006 Neurotechnologija

Remarks

Unused method.

See Also

GetStaticParameter | CopyParameters | constants

7.9.1.4.22. Unregister Method

Unregisters Neurotec.Biometrics.NfsMatcher library.

```
public static void Unregister();
```

See Also

Register | IsRegistered

7.9.1.4.23. Verify Method

Compares two NFTempalte.

7.9.1.4.23.1. Verify(IntPtr,int,IntPtr,int)

```
public int Verify(
        IntPtr template1,
        int template1Size,
        IntPtr template2,
        int template2Size
);
```

Parameters

template1	Pointer to memory block that contains packed NFTemplate.
template1Size	The template size.
template2	Pointer to memory block that contains packed NFTemplate.
template2Size	The template size.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NFTemplate

7.9.1.4.23.2. Verify(IntPtr,int,IntPtr,int,out NfsmMatchDetails)

```
public int Verify(
        IntPtr template1,
        int template1Size,
        IntPtr template2,
        int template2Size,
        out NfsmMatchDetails matchDetails
);
```

Parameters

template1	Pointer to memory block that contains packed NFTemplate.
template1Size	The template size.
template2	Pointer to memory block that contains packed NFTemplate.
template2Size	The template size.
matchDetails	The NfsmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NfsmMatchDetails | NFTemplate

Copyright © 2006 Neurotechnologija

7.9.1.4.23.3. Verify Method

```
public int Verify(
        byte[] template1,
        byte[] template2
);
```

Parameters

template1	The byte array that contains packed NFTemplate.
template2	The byte array that contains packed NFTemplate.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NFTemplate

7.9.1.4.23.4. Verify(byte[],byte[],out NfsmMatchDetails)

```
public int Verify(
        byte[] template1,
        byte[] template2,
        out NfsmMatchDetails matchDetails
);
```

Parameters

template1	The byte array that contains packed NFTemplate.
template2	The byte array that contains packed NFTemplate.
matchDetails	The NfsmMatchDetails object.

Copyright © 2006 Neurotechnologija

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NfsmMatchDetails | NFTemplate

7.9.1.5. NfsmUsedMatcher Enumeration

Specifies the used algorithm.

```
public enum NfsmUsedMatcher
```

Members

Member	Description
NF	NFMatcher algorithm.
NFAndVFFusion	The fusion of both NFMatcher and VFMatcher algorithms.
VF	VFMatcher algorithm.

7.10. Neurotec.Biometrics.NFTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates.

DLL: `Neurotec.Biometrics.NFTemplate.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).

7.10.1. Neurotec.Biometrics Namespace

Provides functionality for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).

Classes

Class	Description
NFTemplate	Provides methods for packing, unpacking and editing Neurotechnologija Fingers Templates (NFTemplates).
NFTemplate.RecordCollection	Represents the collection of NFRecord.
NFTemplateInfo	For internal use.

7.10.1.1. NFTemplate Class

Constructors

NFTemplate Constructor	Initializes a new instance of the `NFTemplate` class.

Properties

Handle	Gets handle to unmanaged `NFTemplate` object.
Records	Gets Records collection.

Methods

CalculateSize	Calculates `NFTemplate` size.
Check	Checks if format of packed `NFTemplate` format is correct.
Clone	Creates `NFTemplate` object from another `NFTemplate` object.
Dispose	Releases the resources used by NFTemplate.
FromHandle	Creates `NFTemplate` object from handle.
GetRecordCount	Retrieves records count.
GetSize	Calculates packed size of `NFTemplate` ob-

Copyright © 2006 Neurotechnologija

	ject.
Pack	Creates `NFTemplate` object with `NFRecord` objects and packs into byte array.
Save	Packs `NFTemplate` object to byte array.
Unpack	Unpacks `NFTemplate` object.

Constants

DllName	Name of DLL containing unmanaged part of this class.
FlagProcessFirstRecordOnly	The flag indicating whether only the first finger record should be unpacked or packed while unpacking or packing NFTemplate.
MaxRecordCount	The maximum number of finger records NFTemplate can contain.

7.10.1.1.1. NFTemplate Constructor

7.10.1.1.1.1. NFTemplate();

```
public NFTemplate()
```

7.10.1.1.1.2. NFTemplate(byte[] buffer)

Creates NFTemplate object from packed NFTemplate object.

```
public NFTemplate(
        byte[] buffer
);
```

Parameters

buffer	A byte array with packed NFTemplate objects.

See Also

Pack | Save

7.10.1.1.1.3. NFTemplate(byte[] buffer, uint flags)

Creates NFTemplate object from packed NFTemplate object. Behavior is controlled through flags.

```
public NFTemplate(
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	A byte array with packed NFTemplate objects.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

See Also

Pack | Save

7.10.1.1.1.4. NFTemplate(byte[] buffer, out NFTemplateInfo info)

Creates NFTemplate object from packed NFTemplate object.

```
public NFTemplate(
        byte[] buffer,
        out NFTemplateInfo info
);
```

Parameters

buffer	A byte array with packed NFTemplate objects.
info	NFTemplateInfo object.

See Also

Pack | Save

7.10.1.1.1.5. NFTemplate(byte[] buffer, uint flags, out NFTemplateInfo info)

Creates NFTemplate object from packed NFTemplate object. Behavior is controlled through flags.

Copyright © 2006 Neurotechnologija

```
public NFTemplate(
        byte[] buffer,
        uint flags,
        out NFTemplateInfo info
);
```

Parameters

buffer	A byte array with packed NFTemplate objects.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
info	NFTemplateInfo object.

See Also

Pack | Save

7.10.1.1.2. Handle Property

Gets handle to unmanaged `NFTemplate` object.

```
public IntPtr Handle {get;}
```

Property value

A handle to unmanaged `NFTemplate` object.

See Also

FromHandle

7.10.1.1.3. Records Property

Gets Records collection.

```
public NFTemplate.RecordCollection Records {get;}
```

Property value

A NFTemplate.RecordCollection collection.

See Also

NFTemplate.RecordCollection | NFRecord

7.10.1.1.4. CalculateSize Method

Calculates `NFTemplate` size.

```
public static int CalculateSize(
        int[] recordSizes
);
```

Parameters

recordSizes	An array that contains `NFRecords` sizes.

Return Values

A size of records.

See Also

NFRecord

7.10.1.1.5. Check Method

Checks if format of packed `NFTemplate` format is correct.

7.10.1.1.5.1. Check(byte[] buffer)

Checks if format of packed `NFTemplate` format is correct.

```
public static void Check(
        byte[] buffer
);
```

Parameters

buffer	A byte array with packed `NFTemplate`.

Return Values

`true` if `NFTemplate` format is correct, `false` otherwise.

7.10.1.1.5.2. Check(IntPtr buffer, int bufferSize)

Checks if format of packed `NFTemplate` format is correct.

```
public static void Check(
        IntPtr buffer,
```

```
        int bufferSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFTemplate`.
bufferSize	Size of memory buffer that contains packed `NFTemplate`.

Return Values

`true` if `NFTemplate` format is correct, `false` otherwise.

7.10.1.1.6. Clone Method

Creates `NFTemplate` object from another `NFTemplate` object.

```
public object Clone();
```

Return Values

A NFTemplate object.

See Also

NFTemplate

7.10.1.1.7. Dispose Method

Releases the resources used by NFTemplate.

```
public void Dispose();
```

See Also

NFTemplate

7.10.1.1.8. FromHandle Method

Creates `NFTemplate` object from handle.

```
public static NFTemplate FromHandle(
        IntPtr handle
);
```

Copyright © 2006 Neurotechnologija

Parameters

handle	Handle to unmanaged `NFTemplate` object.

Return Values

A NFTemplate object.

See Also

NFTemplate | Handle

7.10.1.1.9. GetRecordCount Method

Retrieves records count.

7.10.1.1.9.1. GetRecordCount(byte[] buffer)

Retrieves records count.

```
public static int GetRecordCount(
        byte[] buffer
);
```

Parameters

buffer	A byte array with `NFTemplate`.

Return Values

Records count.

See Also

NFRecord

7.10.1.1.9.2. GetRecordCount(IntPtr buffer, int bufferSize)

Retrieves records count.

```
public static int GetRecordCount(
        IntPtr buffer,
        int bufferSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFTemplate`.
bufferSize	Size of memory buffer that contains packed `NFTemplate`.

Return Values

Records count.

See Also

NFRecord

7.10.1.1.10. GetSize Method

Calculates packed size of `NFTemplate` object.

7.10.1.1.10.1. GetSize()

Calculates packed size of `NFTemplate` object.

```
public int GetSize();
```

Return Values

Size of `NFTemplate` object.

7.10.1.1.10.2. GetSize(uint flags)

Calculates packed size of `NFTemplate` object. Behavior is controlled through flags.

```
public int GetSize(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

Size of `NFTemplate` object.

7.10.1.1.11. Pack Method

Creates `NFTemplate` object with `NFRecord` objects and packs into byte array.

7.10.1.1.11.1. Pack(params byte[][] records)

Packs `NFRecord` objects into byte array.

```
public static byte[] Pack(
        params byte[][] records
);
```

Parameters

records	A byte array of `NFRecord` objects byte arrays.

Return Values

A byte array with packed `NFTemplate` objects.

See Also

NFRecord | Save

7.10.1.1.11.2. Pack(IntPtr[] records, int[] recordSizes)

Packs `NFRecord` objects into byte array from memory buffer.

```
public static byte[] Pack(
        IntPtr[] records,
        int[] recordSizes
);
```

Parameters

records	Pointer to NFRecord objects array.
recordSizes	An array that contains NFRecord objects sizes.

Return Values

A byte array with packed `NFTemplate` object.

See Also

NFRecord | Save

7.10.1.1.12. Save Method

Packs `NFTemplate` object to byte array.

7.10.1.1.12.1. Save()

Saves `NFTemplate` object to byte array.

```
public byte[] Save();
```

Return Values

A byte array with packed `NFTemplate` object.

See Also

Pack

7.10.1.1.12.2. Save(uint flags)

Packs `NFTemplate` object to byte array. Behavior is controlled through flags.

```
public byte[] Save(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that control behavior of the method.

Return Values

A byte array with packed `NFTemplate` object.

See Also

Pack

7.10.1.1.13. Unpack Method

Unpacks `NFTemplate` object.

7.10.1.1.13.1. int Unpack(IntPtr buffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr[] records,

Copyright © 2006 Neurotechnologija

out int[] recordSizes)

Unpacks `NFTemplate` object from memory buffer.

```
public static int Unpack(
        IntPtr buffer,
        int bufferSize,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out IntPtr[] records,
        out int[] recordSizes
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFTemplate` object.
bufferSize	Size of memory buffer that contains packed `NFTemplate`.
majorVersion	Major version of `NFTemplate`.
minorVersion	Minor version of `NFTemplate`.
size	The size of buffer from header of packed NFTemplate. This size is equal to buffer-Size.
headerSize	The size of packed NFTemplate header size.
records	The array of NFRecords.
recordSizes	An array that contains `NFRecord` sizes.

Return Values

The recourds count.

See Also

NFRecord | Pack

7.10.1.1.13.2. void Unpack(IntPtr buffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[][] records)

Unpacks `NFTemplate` object from memory buffer.

```
public static void Unpack(
        IntPtr buffer,
        int bufferSize,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out byte[][] records
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NFTemplate` object.
bufferSize	Size of memory buffer that contains packed `NFTemplate`.
majorVersion	Major version of `NFTemplate`.
minorVersion	Minor version of `NFTemplate`.
size	The size of buffer from header of packed NFTemplate. This size is equal to buffer-Size.
headerSize	The size of packed NFTemplate header size.
records	A byte array that contains `NFRecord` objects bytes arrays.

See Also

NFRecord | Pack

7.10.1.1.13.3. int Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr[] records, out int[] recordSizes)

Unpacks `NFTemplate` object from byte array.

```
public static int Unpack(
        byte[] buffer,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out IntPtr[] records,
        out int[] recordSizes
);
```

Parameters

buffer	Byte array that contains packed `NFTemplate` object.
majorVersion	Major version of `NFTemplate`.
minorVersion	Minor version of `NFTemplate`.
size	The size of buffer from header of packed NFTemplate. This size is equal to bufferSize.
headerSize	The size of packed NFTemplate header size.
records	The array of NFRecords.
recordSizes	An array that contains `NFRecord` sizes.

Return Values

The recourds count.

See Also

NFRecord | Pack | Save

7.10.1.1.13.4. void Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[][] records)

Unpacks `NFTemplate` object from byte array.

```
public static void Unpack(
        byte[] buffer,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out byte[][] records
);
```

Parameters

buffer	Byte array that contains packed `NFTemplate` object.
majorVersion	Major version of `NFTemplate`.
minorVersion	Minor version of `NFTemplate`.

Copyright © 2006 Neurotechnologija

size	The size of buffer from header of packed NFTemplate. This size is equal to buffer-Size.
headerSize	The size of packed NFTemplate header size.
records	A byte array that contains `NFRecord` objects bytes arrays.

See Also

NFRecord | Pack | Save

7.10.1.2. NFTemplate.RecordCollection Class

Properties

Capacity	Gets or sets the number of elements that the RecordCollection can contain.
Item	Gets the member from collection by index.

Methods

Add	Adds an object to the end of the RecordCollection.
AddCopy	Copies NFRecord object to the NFTemplate.RecordCollection.

7.10.1.2.1. Capacity Property

Gets or sets the number of elements that the RecordCollection can contain.

```
public int Capacity {get; set;}
```

Property value

The number of elements that the `NFTemplate.RecordCollection` can contain.

7.10.1.2.2. NFTemplate.RecordCollection.Item Property

Gets the member from collection by index.

```
public NFRecord this[
```

```
        int index
] {get;}
```

Property value

A NFRecord object.

7.10.1.2.3. Add Methods

Adds an object to the end of the RecordCollection.

7.10.1.2.3.1. Add(byte[] buffer)

```
public NFRecord Add(
        byte[] buffer
);
```

Parameters

buffer	The packed NFRecord object.

Return Values

The added to the collection NFRecord object.

See Also

NFRecord

7.10.1.2.3.2. Add(byte[] buffer, uint flags)

```
public NFRecord Add(
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	The packed NFRecord object.
flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The added to the collection NFRecord object.

See Also

NFRecord

7.10.1.2.3.3. Add(ushort width, ushort height, ushort horzResolution, ushort vertResolution)

```
public NFRecord Add(
        ushort width,
        ushort height,
        ushort horzResolution,
        ushort vertResolution
);
```

Parameters

width	The width of fingerprint image.
height	The height of fingerprint image.
horzResolution	The horizontal resolution of fingerprint image.
vertResolution	The vertical resolution of fingerprint image.

Return Values

The added to the collection NFRecord object.

See Also

NFRecord

7.10.1.2.3.4. Add(ushort,ushort,ushort,ushort,uint)

```
public NFRecord Add(
        ushort width,
        ushort height,
        ushort horzResolution,
        ushort vertResolution,
        uint flags
);
```

Parameters

width	The width of fingerprint image.
height	The height of fingerprint image.
horzResolution	The horizontal resolution of fingerprint image.
vertResolution	The vertical resolution of fingerprint image.
flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The added to the collection NFRecord object.

See Also

NFRecord

7.10.1.2.4. AddCopy Method

Copies NFRecord object to the NFTemplate.RecordCollection.

```
public NFRecord AddCopy(
        NFRecord srcRecord
);
```

Parameters

srcRecord	The NFRecord object.

Return Values

The copied to the collection NFRecord object.

See Also

NFRecord

7.11. Neurotec.Biometrics.NMatcher Library

Provides functionality for comparing Neurotechnologija Templates.

DLL: `Neurotec.Biometrics.NMatcher.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for comparing Neurotechnologija Templates.

7.11.1. Neurotec.Biometrics Namespace

Provides functionality for comparing Neurotechnologija Templates.

Classes

NMatcher	Provides functionality for comparing Neurotechnologija Templates (NTemplates), encapsulated in Neurotechnologija Matcher (NMatcher) object.
NMMatchDetails	Represents information about fingers match details.

7.11.1.1. NMatcher Class

Provides functionality for comparing Neurotechnologija Templates (NTemplates), encapsulated in Neurotechnologija Matcher (NMatcher) object.

Constructors

NMatcher Constructor	Initializes a new instance of the `NMatcher` class.

Properties

FingersNfmMatchingSpeed	Gets or sets matching speed.
FingersNfmMaximalRotation	Gets or sets maximal rotation.
FingersNfmMode	Gets or sets scanners mode.
FingersUsedMatcher	Gets or sets used matching algorithm.
FingersVfmMatchingSpeed	Gets or sets matching speed.
FingersVfmMaximalRotation	Gets or sets maximal rotation.
FingersVfmMode	Gets or sets scanners mode.
IsRegistered	Gets a value indicating whether Neuro-

Copyright © 2006 Neurotechnologija

	tec.Biometrics.NMatcher library is registered.
MatchingThreshold	Gets or sets the matching threshold.

Methods

CopyParameters	Copies parameters values from one NMatcher object to another NMatcher object.
Dispose	Releases the resources used by `NMatcher`.
GetParameter	Gets value of parameter by parameter id.
GetStaticParameter	Gets value of static parameter by parameter id.
IdentifyEnd	Ends identification process.
IdentifyNext	Compares the specified NTemplate with the one identification was started with.
IdentifyStart	Starts identification with the specified packed NTemplate.
Register	Registers Neurotec.Biometrics.NMatcher library.
Reset	Resets all NMatcher parameters to default values.
SetParameter	Sets value of parameter by parameter id.
SetStaticParameter	Sets value of static parameter by parameter id.
Unregister	Unregisters Neurotec.Biometrics.NMatcher library.
Verify	Compares two NTemplate.

Constants

DllName	Name of DLL containing unmanaged part of this class.
ParameterCopyright	Identifier specifying library copyright static read-only parameter of type string.

Copyright © 2006 Neurotechnologija

ParameterMatchingThreshold	Identifier specifying matching threshold (biggest allowed FAR) parameter of type int. Parameter value is equal to - 12 * log10(FAR) and must be not less than zero (for example, 48 for FAR = 0.01%).
ParameterName	Identifier specifying library name static read-only parameter of type string.
ParameterVersionHigh	Identifier specifying high part of library version static read-only parameter of type uint. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
ParameterVersionLow	Identifier specifying low part of library version static read-only parameter of type uint. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.
PartFingers	Identifier specifying that retrieving or setting parameter for fingers matcher (NfsMatcher) contained in NMatcher.
PartNone	Identifier specifying that retrieving or setting parameter for NMatcher.

7.11.1.1.1. NMatcher Constructor

Initializes a new instance of the NMatcher class.

```
public NMatcher();
```

7.11.1.1.2. FingersNfmMatchingSpeed Property

Gets or sets matching speed.

```
public NfmSpeed FingersNfmMatchingSpeed {get; set;}
```

Property value

One of the NfmSpeed values. The default is `NfmSpeed.Low`.

See Also

NfmSpeed

7.11.1.1.3. FingersNfmMaximalRotation Property

Gets or sets maximal rotation.

```
public byte FingersNfmMaximalRotation {get; set;}
```

Property value

The finger maximal rotation.

7.11.1.1.4. FingersNfmMode Property

Gets or sets scanners mode.

```
public uint FingersNfmMode {get; set;}
```

Property value

On of the scanner types.

See Also

constants

7.11.1.1.5. FingersUsedMatcher Property

Gets or sets used matching algorithm.

```
public NfsmUsedMatcher FingersUsedMatcher {get; set;}
```

Property value

One of the NfsmUsedMatcher values. The default is `NfsmUsedMatcher.NF`

See Also

NfsmUsedMatcher

7.11.1.1.6. FingersVfmMatchingSpeed Property

Gets or sets matching speed.

```
public VfmSpeed FingersVfmMatchingSpeed {get; set;}
```

Property value

One of the VfmSpeed values. The default is `VfmSpeed.Low`.

Copyright © 2006 Neurotechnologija

See Also

VfmSpeed

7.11.1.1.7. FingersVfmMaximalRotation Property

Gets or sets maximal rotation.

```
public byte FingersVfmMaximalRotation {get; set;}
```

Property value

See Also

ParameterMaximalRotation

7.11.1.1.8. FingersVfmMode Property

Gets or sets scanners mode.

```
public uint FingersVfmMode {get; set;}
```

Property value

One of the scanner types.

See Also

constants

7.11.1.1.9. IsRegistered Property

Gets a value indicating whether Neurotec.Biometrics.NMatcher library is registered.

```
public static bool IsRegistered {get;}
```

Property value

`true` if Neurotec.Biometrics.NMatcher library is registered; otherwise, `false`.

See Also

Register | Unregister

7.11.1.1.10. MatchingThreshold Property

Gets or sets the matching threshold.

Copyright © 2006 Neurotechnologija

```
public int MatchingThreshold {get; set;}
```

Property value

The matching speed.

See Also

ParameterMatchingThreshold

7.11.1.1.11. CopyParameters Method

Copies parameters values from one NMatcher object to another NMatcher object.

```
public static void CopyParameters(
        NMatcher sourceMatcher,
        NMatcher destinationMatcher
);
```

Parameters

sourceMatcher	The NMatcher object.
destinationMatcher	The NMatcher object.

See Also

Constants

7.11.1.1.12. Dispose Method

Releases the resources used by `NMatcher`.

```
public void Dispose();
```

7.11.1.1.13. GetParameter Method

Gets value of parameter by parameter id.

```
public object GetParameter(
        ushort partId,
        ushort parameterId
);
```

Parameters

partId	Specifies the matcher.
parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

partId can be one of the following:

- PartFingers + any *partId* NfsMatcher GetParameter method supports
- PartNone

If *partId* is PartFingers + any *partId* NfsMatcher GetParameter method supports, *parameterId* must be any NfsMatcher GetParameter method supports except for ParameterMatchingThreshold.

If *partId* is PartNone, the following values can be used for *parameterId*:

- ParameterMatchingThreshold

See Also

SetParameter | CopyParameters

7.11.1.1.14. GetStaticParameter Method

Gets value of static parameter by parameter id.

```
public static object GetStaticParameter(
        ushort partId,
        ushort parameterId
);
```

Parameters

partId	Specifies the matcher.
parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

partId can be one of the following:

- PartFingers + any *partId* GetStaticParameter method supports
- PartNone

If *partId* is PartFingers + any *partId* NfsMatcher GetStaticParameter method supports, *parameterId* must be any NfsMatcher GetStaticParameter method supports.

If *partId* is PartNone, the following values can be used for *parameterId*:

- ParameterCopyright
- ParameterName
- ParameterVersionHigh
- ParameterVersionLow

See Also

SetStaticParameter | CopyParameters

7.11.1.1.15. IdentifyEnd Method

Ends identification process.

```
public void IdentifyEnd();
```

Remarks

See Also

IdentifyStart | IdentifyNext

7.11.1.1.16. IdentifyNext Method

Compares the specified NTemplate with the one identification was started with.

7.11.1.1.16.1. IdentifyNext(IntPtr,int)

```
public int IdentifyNext(
        IntPtr template,
        int templateSize
);
```

Parameters

template	Pointer to memory block that contains packed NTemplate.
templateSize	The template size.

Return Values

The score of matched NFTemplates.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NTemplate

7.11.1.1.16.2. IdentifyNext(IntPtr,int,NMMatchDetails)

```
public int IdentifyNext(
        IntPtr template,
        int templateSize,
        NMMatchDetails matchDetails
);
```

Parameters

template	Pointer to memory block that contains packed NTemplate.
templateSize	The template size.
matchDetails	The NMMatchDetails object.

Return Values

The score of matched NFTemplates.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NMMatchDetails | NTemplate

7.11.1.1.16.3. IdentifyNext(byte[])

```
public int IdentifyNext(
        byte[] template
);
```

Parameters

template	The byte array that contains packed NTemplate.

Return Values

The score of matched NFTemplates.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NTemplate

7.11.1.1.16.4. IdentifyNext(byte[],NMMatchDetails)

```
public int IdentifyNext(
        byte[] template,
        NMMatchDetails matchDetails
);
```

Parameters

template	The byte array that contains packed NTemplate.
matchDetails	The NMMatchDetails object.

Return Values

Copyright © 2006 Neurotechnologija

The score of matched NFTemplates.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NMMatchDetails | NTemplate

7.11.1.1.17. IdentifyStart Method

Starts identification with the specified packed NTemplate.

7.11.1.1.17.1. IdentifyStart(IntPtr template, int templateSize)

```
public void IdentifyStart(
        IntPtr template,
        int templateSize
);
```

Parameters

template	Pointer to memory block that contains packed NTemplate.
templateSize	The template size.

See Also

IdentifyNext | IdentifyEnd | NTemplate

7.11.1.1.17.2. IdentifyStart(IntPtr,int,out NMMatchDetails)

```
public void IdentifyStart(
        IntPtr template,
        int templateSize,
        out NMMatchDetails matchDetails
);
```

Parameters

template	Pointer to memory block that contains

	packed NTemplate.
templateSize	The template size.
matchDetails	The NMMatchDetails object.

See Also

IdentifyNext | IdentifyEnd NMMatchDetails | NTemplate

7.11.1.1.17.3. IdentifyStart(byte[])

```
public void IdentifyStart(
        byte[] template
);
```

Parameters

template	The byte array with packed NFTemplate.

See Also

IdentifyNext | IdentifyEnd | NTemplate

7.11.1.1.17.4. IdentifyStart(byte[],out NMMatchDetails)

```
public void IdentifyStart(
        byte[] template,
        out NMMatchDetails matchDetails
);
```

Parameters

template	The byte array with packed NFTemplate.
matchDetails	The NMMatchDetails object.

See Also

IdentifyNext | IdentifyEnd | NMMatchDetails | NTemplate

7.11.1.1.18. Register Method

Registers Neurotec.Biometrics.NMatcher library.

```
public static void Register(
        string server
);
```

Parameters

server	The string that specifies license manager server name or IP address.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

7.11.1.1.19. Reset Method

Resets all NMatcher parameters to default values.

```
public void Reset();
```

7.11.1.1.20. SetParameter Method

Sets value of parameter by parameter id.

```
public void SetParameter(
        ushort partId,
        ushort parameterId,
        object value
);
```

Parameters

partId	Specifies the matcher.
parameterId	The parameter identifier.
value	The parameter value.

Remarks

partId can be one of the following:

- PartFingers + any *partId* NfsMatcher SetParameter method supports
- PartNone

If *partId* is PartFingers + any *partId* NfsMatcher SetParameter method supports, *parameterId* must be any NfsMatcher SetParameter method supports except for ParameterMatchingThreshold.

If *partId* is PartNone, the following values can be used for *parameterId*:

- ParameterMatchingThreshold

See Also

GetParameter | CopyParameters

7.11.1.1.21. SetStaticParameter Method

Sets value of static parameter by parameter id.

```
public static void SetStaticParameter(
        ushort partId,
        ushort parameterId,
        object value
);
```

Parameters

partId	Specifies the matcher.
parameterId	The parameter identifier.
value	The parameter value.

Remarks

Unused method.

See Also

GetStaticParameter | CopyParameters

7.11.1.1.22. Unregister Method

Unregisters Neurotec.Biometrics.NMatcher library.

```
public static void Unregister();
```

See Also

Register | IsRegistered

7.11.1.1.23. Verify Method

Compares two NTemplate.

7.11.1.1.23.1. Verify(IntPtr,int,IntPtr,int)

```
public int Verify(
        IntPtr template1,
        int template1Size,
        IntPtr template2,
        int template2Size
);
```

Parameters

template1	Pointer to memory block that contains packed NTemplate.
template1Size	The template size.
template2	Pointer to memory block that contains packed NTemplate.
template2Size	The template size.

Return Values

The score of matched NFTemplates.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NTemplate

7.11.1.1.23.2. Verify(IntPtr,int,IntPtr,int,out NMMatchDetails)

```
public int Verify(
        IntPtr template1,
        int template1Size,
        IntPtr template2,
        int template2Size,
        out NMMatchDetails matchDetails
);
```

Parameters

template1	Pointer to memory block that contains packed NTemplate.
template1Size	The template size.
template2	Pointer to memory block that contains packed NTemplate.
template2Size	The template size.
matchDetails	The NMMatchDetails object.

Return Values

The score of matched NFTemplates.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NMMatchDetails NTemplate

7.11.1.1.23.3. Verify(byte[],byte[])

```
public int Verify(
        byte[] template1,
        byte[] template2
);
```

Parameters

template1	The byte array that contains packed NTemplate.

template2	The byte array that contains packed NTemplate.

Return Values

The score of matched NFTemplates.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NTemplate

7.11.1.1.23.4. Verify(byte[],byte[],out NMMatchDetails)

```
public int Verify(
        byte[] template1,
        byte[] template2,
        out NMMatchDetails matchDetails
);
```

Parameters

template1	The byte array that contains packed NTemplate.
template2	The byte array that contains packed NTemplate.
matchDetails	The NMMatchDetails object.

Return Values

The score of matched NFTemplates.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NTemplates do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NMMatchDetails NTemplate

7.11.1.2. NMMatchDetails Class

Represents information about fingers match details.

Properties

FingersMatchDetails	Gets details of fingers matching.
Score	Specifies score of matched NTemplate templates

Methods

Dispose	Releases the resources used by NMMatchDetails.

7.11.1.2.1. FingersMatchDetails Property

Gets details of fingers matching.

```
public NfsmMatchDetails FingersMatchDetails {get;}
```

Property value

A NfsmMatchDetails that represents the details of finger matching.

See Also

NfsmMatchDetails

7.11.1.2.2. Score Property

Specifies score of matched NTemplate templates

```
public int Score {get;}
```

Property value

The score of matched NTemplate templates.

See Also

NTemplate

Copyright © 2006 Neurotechnologija

7.11.1.2.3. Dispose Method

Releases the resources used by NMMatchDetails.

```
public void Dispose();
```

7.12. Neurotec.Biometrics.NTemplate Library

Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

DLL: `Neurotec.Biometrics.NTemplate.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

7.12.1. Neurotec.Biometrics Namespace

Provides functionality for packing, unpacking and editing Neurotechnologija Templates.

Classes

Class	Description
NTemplate	Provides functionality for packing, unpacking and editing Neurotechnologija Templates (NTemplates).
NTemplateInfo	For internal use.

7.12.1.1. NTemplate Class

Constructors

NTemplate Constructor	Initializes a new instance of the `NTemplate` class.

Properties

Fingers	Gets NFTemplate object.

Handle	Gets handle to unmanaged `NTemplate` object.

Methods

AddFingers	Creates new NFTemplate object.
AddFingersCopy	Creates one NFTemplate object from another.
CalculateSize	Calculates `NTemplate` size.
Check	Checks if format of packed `NTemplate` format is correct.
Clear	Removes all `NFRecord` objects.
Clone	Creates `NTemplate` object from another `NTemplate` object.
Dispose	Releases the resources used by `NTemplate`.
FromHandle	Creates `NTemplate` object from handle.
GetSize	Calculates packed size of NFTemplate object.
Pack	Packs packed fingers template as NTemplate into the specified memory buffer.
RemoveFingers	Removes fingers template from the NTemplate.
Save	Packs `NTemplate` object to byte array.
Unpack	Unpacks packed fingers template from the packed NTemplate.

Constants

DllName	Name of DLL containing unmanaged part of this class.

7.12.1.1.1. NTemplate Constructor

Copyright © 2006 Neurotechnologija

Initializes a new instance of the `NTemplate` class.

7.12.1.1.1.1. NTemplate()

Initializes a new instance of the `NTemplate` class.

```
public NTemplate();
```

7.12.1.1.1.2. NTemplate(byte[] buffer)

Initializes a new instance of the `NTemplate` class from byte array.

```
public NTemplate(
        byte[] buffer
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.

See Also

Pack | Save

7.12.1.1.1.3. NTemplate(byte[] buffer, uint flags)

Initializes a new instance of the `NTemplate` class from byte array. Behavior is controlled through flags.

```
public NTemplate(
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.

See Also

Pack | Save

7.12.1.1.1.4. NTemplate(byte[] buffer, out NTemplateInfo info)

Initializes a new instance of the `NTemplate` class from byte array.

```
public NTemplate(
        byte[] buffer,
        out NTemplateInfo info
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.
info	The `FMRecordInfo` object.

See Also

Pack | Save

7.12.1.1.1.5. NTemplate(byte[] buffer, uint flags, out NTemplateInfo info)

Initializes a new instance of the `NTemplate` class from byte array. Behavior is controlled through flags.

```
public NTemplate(
        byte[] buffer,
        uint flags,
        out NTemplateInfo info
);
```

Parameters

buffer	Byte array that contains packed `NTemplate`.
flags	Bitwise combination of zero or more flags that controls behavior of the constructor.
info	The `FMRecordInfo` object.

See Also

Pack | Save

7.12.1.1.2. Fingers Property

Gets NFTemplate object.

```
public NFTemplate Fingers {get;}
```

Property value

A NFTemplate object.

See Also

NFTemplate

7.12.1.1.3. Handle Property

Gets handle to unmanaged `NTemplate` object.

```
public IntPtr Handle {get;}
```

Property value

A handle to unmanaged `NTemplate` object.

See Also

FromHandle

7.12.1.1.4. AddFingers Methods

Creates new NFTemplate object.

7.12.1.1.4.1. AddFingers()

```
public NFTemplate AddFingers();
```

Return Values

A NFTemplate object.

See Also

NFTemplate

7.12.1.1.4.2. AddFingers(byte[] buffer)

Creates new NFTemplate object from packed NFTemplate object.

Copyright © 2006 Neurotechnologija

```
public NFTemplate AddFingers(
        byte[] buffer
);
```

Parameters

buffer	The packed NFTemplate object.

Return Values

The NFTemplate object.

See Also

NFTemplate

7.12.1.1.4.3. AddFingers(byte[] buffer, uint flags)

Creates new NFTemplate object from packed NFTemplate object.

```
public NFTemplate AddFingers(
        byte[] buffer,
        uint flags
);
```

Parameters

buffer	The packed NFTemplate object.
flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

The NFTemplate object.

See Also

NFTemplate

7.12.1.1.5. AddFingersCopy Method

Creates one NFTemplate object from another.

```
public NFTemplate AddFingersCopy(
        NFTemplate srcFingers
);
```

Parameters

srcFingers	The NFTemplate object.

Return Values

The NFTemplate object.

See Also

NFTemplate

7.12.1.1.6. CalculateSize Method

Calculates `NTemplate` size.

```
public static int CalculateSize(
        int fingersTemplateSize
);
```

Parameters

fingersTemplateSize	Size of packed NFTemplate.

Return Values

The size of packed NTemplate.

See Also

NFTemplate | Pack

7.12.1.1.7. Check

Checks if format of packed `NTemplate` format is correct.

7.12.1.1.7.1. Check(byte[] buffer)

Checks if format of packed `NTemplate` format is correct.

```
public static void Check(
        byte[] buffer
);
```

Parameters

buffer	Byte array that contains packed NTemplate.

Return Values

true if NTemplate format is correct; false otherwise.

See Also

Pack | Save

7.12.1.1.7.2. Check(IntPtr buffer, int bufferSize)

Checks if format of packed NTemplate format is correct.

```
public static void Check(
        IntPtr buffer,
        int bufferSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed NTemplate.
bufferSize	Size of memory buffer that contains packed NTemplate.

Return Values

true if NTemplate format is correct; false otherwise.

See Also

Pack | Save

7.12.1.1.8. Clear Method

Removes all NFRecord objects.

```
public void Clear();
```

See Also

NFRecord

7.12.1.1.9. Clone Method

Creates `NTemplate` object from another `NTemplate` object.

```
public virtual object Clone();
```

Return Values

A `NTemplate` object.

7.12.1.1.10. Dispose Method

Releases the resources used by `NTemplate`.

```
public void Dispose();
```

7.12.1.1.11. FromHandle Method

Creates `NTemplate` object from handle.

```
public static NTemplate FromHandle(
        IntPtr handle
);
```

Parameters

handle	Handle to unmanaged `NTemplate` object.

Return Values

A NTemplate object.

See Also

Handle

7.12.1.1.12. GetSize Methods

Calculates packed size of NFTemplate object.

7.12.1.1.12.1. GetSize()

Calculates packed size of NFTemplate object.

```
public int GetSize();
```

Return Values

Size of `NTemplate` object.

7.12.1.1.12.2. GetSize(uint flags)

Calculates packed size of NFTemplate object. Behavior is controlled through flags.

```
public int GetSize(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that controls behavior of the method.

Return Values

Size of `NTemplate` object.

7.12.1.1.13. Pack Methods

Packs packed fingers template as NTemplate into the specified memory buffer.

7.12.1.1.13.1. Pack(byte[] fingersTemplate)

```
public static byte[] Pack(
        byte[] fingersTemplate
);
```

Parameters

fingersTemplate	The byte array with packed NFTemplate.

Return Values

A byte array with packed `NTemplate` objects.

See Also

NFTemplate | Save | Unpack

7.12.1.1.13.2. Pack(IntPtr fingersTemplate, int fingersTemplateSize)

```
public static byte[] Pack(
        IntPtr fingersTemplate,
        int fingersTemplateSize
);
```

Parameters

fingersTemplate	Pointer to memory buffer.
fingersTemplateSize	The fingers template size.

Return Values

A byte array with packed `NTemplate` objects.

See Also

NFTemplate | Save | Unpack

7.12.1.1.14. RemoveFingers Method

Removes fingers template from the NTemplate.

```
public void RemoveFingers();
```

See Also

NFTemplate | AddFingers | AddFingersCopy

7.12.1.1.15. Save Methods

Packs `NTemplate` object to byte array.

7.12.1.1.15.1. Save()

Packs `NTemplate` object to byte array.

```
public byte[] Save();
```

Return Values

A byte array with packed `NTemplate` object.

See Also

Pack

7.12.1.1.15.2. Save(uint flags)

Packs `NTemplate` object to byte array. Behavior is controlled through flags.

```
public byte[] Save(
        uint flags
);
```

Parameters

flags	Bitwise combination of zero or more flags that control behavior of the method.

Return Values

A byte array with packed `NTemplate` object.

See Also

Pack

7.12.1.1.16. Unpack Methods

Unpacks packed fingers template from the packed NTemplate.

7.12.1.1.16.1. Unpack(IntPtr buffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr fingersTemplate, out int fingersTemplateSize)

Unpacks `NTemplate` object from memory buffer.

```
public static void Unpack(
        IntPtr buffer,
        int bufferSize,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out IntPtr fingersTemplate,
        out int fingersTemplateSize
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NTemplate` object.
bufferSize	Size of memory buffer that contains packed

	`NTemplate`.
majorVersion	Major version of `NTemplate`.
minorVersion	Minor version of `NTemplate`.
size	The size of buffer from header of packed NTemplate. This size is equal to bufferSize
headerSize	The size of packed NTemplate header size.
fingersTemplate	Pointer to array that contains packed NFTemplate.
fingersTemplateSize	The size of packed NFTemplate.

See Also

NFTemplate | Pack | Save

7.12.1.1.16.2. Unpack(IntPtr buffer, int bufferSize, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out byte[] fingersTemplate)

Unpacks `NTemplate` object from memory buffer.

```
public static void Unpack(
        IntPtr buffer,
        int bufferSize,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out byte[] fingersTemplate
);
```

Parameters

buffer	Pointer to memory buffer that contains packed `NTemplate` object.
bufferSize	Size of memory buffer that contains packed `NTemplate`.
majorVersion	Major version of `NTemplate`.
minorVersion	Minor version of `NTemplate`.
size	The size of buffer from header of packed NTemplate. This size is equal to bufferSize

Copyright © 2006 Neurotechnologija

headerSize	The size of packed NTemplate header size.
fingersTemplate	The byte array with packed NFTemplate.

See Also

NFTemplate | Pack | Save

7.12.1.1.16.3. Unpack(byte[] buffer, out byte majorVersion, out byte minorVersion, out uint size, out byte headerSize, out IntPtr fingersTemplate, out int fingersTemplateSize)

Unpacks `NTemplate` object from byte array.

```
public static void Unpack(
        byte[] buffer,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out IntPtr fingersTemplate,
        out int fingersTemplateSize
);
```

Parameters

buffer	A byte array that contains packed `NTemplate`.
majorVersion	Major version of `NTemplate`.
minorVersion	Minor version of `NTemplate`.
size	The size of buffer from header of packed NTemplate. This size is equal to bufferSize
headerSize	The size of packed NTemplate header size.
fingersTemplate	Pointer to array that contains packed NFTemplate.
fingersTemplateSize	The size of packed NFTemplate.

See Also

NFTemplate | Pack | Save

7.12.1.1.16.4. Unpack(byte[] buffer, out byte majorVersion, out byte minorVer-

sion, out uint size, out byte headerSize, out byte[] fingersTemplate)

Unpacks `NTemplate` object from byte array.

```
public static void Unpack(
        byte[] buffer,
        out byte majorVersion,
        out byte minorVersion,
        out uint size,
        out byte headerSize,
        out byte[] fingersTemplate
);
```

Parameters

buffer	A byte array that contains packed `NTemplate`.
majorVersion	Major version of `NTemplate`.
minorVersion	Minor version of `NTemplate`.
size	The size of buffer from header of packed NTemplate. This size is equal to bufferSize
headerSize	The size of packed NTemplate header size.
fingersTemplate	The byte array with packed NFTemplate.

See Also

NFTemplate | Pack | Save

7.13. Neurotec.Biometrics.VFExtractor Library

Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using VeriFinger algorithm.

DLL: `Neurotec.Biometrics.VFExtractor.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using VeriFinger algorithm.

7.13.1. Neurotec.Biometrics Namespace

Provides functionality for extracting Neurotechnologija Finger Records from fingerprint images using VeriFinger algorithm.

Classes

Class	Description
VFExtractor	Provides methods for extracting Neurotechnologija Finger Records (NFRecords) from fingerprint images using VeriFinger algorithm encapsulated in Neurotechnologija Fingerprint Features Extractor VF (VFExtractor) object.

Enumerations

Enumeration	Description
VfeReturnedImage	Specifies the returned image.

7.13.1.1. VFExtractor Class

Wrapper for MegaMatcher VFExtractor module is implemented in `Neurotec.Biometrics.VFExtractor.dll`.

Constructors

VFExtractor Constructor	Initializes a new instance of the `VFExtractor` class.

Properties

IsRegistered	Gets a value indicating whether VFExtractor library has been registered.
Mode	Gets or sets scanners mode.
ReturnedImage	Gets or sets the image type.

Methods

Copyright © 2006 Neurotechnologija

CopyParameters	Copies the parameters values from one VFExtractor object to another.
Dispose	Releases the resources used by VFExtractor.
Extract	Extracts features from a fingerprint image.
ExtractUnpacked	Extracts features from a fingerprint image and do not packs.
GetMaxTemplateSize	Retrieves maximal size of packed NFRecord the specified VFExtractor can extract.
GetParameter	Retrieves parameter by parameter identifier.
GetStaticParameter	Retrieves static parameter by parameter identifier.
Register	Registers Neurotec.Biometrics.VFExtractor library.
Reset	Resets all parameters of VFExtractor object to default values.
SetParameter	Sets the parameter by parameter identifier.
SetStaticParameter	Sets the static parameter by parameter identifier.
Unregister	Unregisters Neurotec.Biometrics.VFExtractor library.

Constants

DllName	Name of DLL containing unmanaged part of this class.
ParameterMode	Identifier specifying mode (parameter value set) parameter of type uint. Parameter value can be one of the ModeXXX.
ModeAtmelFingerchip ModeAuthentecAes4000 ModeAuthentecAFS2 ModeBiometrikaFX2000 ModeBmfBlp100 ModeCrossmatchVerifier300 ModeDigitalPersonaUareU ModeEthentica	Represents scanners.

Copyright © 2006 Neurotechnologija

ModeGeneral ModeIdenticatorTechnologyDF90 ModeIdentixTouchView ModeKeytronicSecureDesktop ModePreciseBiometrics100CS ModeSecugenHamster ModeSecugenHamster	
ParameterCopyright	Identifier specifying library copyright static read-only parameter of type string.
ParameterName	Identifier specifying library name static read-only parameter of type string.
ParameterReturnedImage	Identifier specifying kind of image returned after extraction parameter of type int. Parameter value can be one of the VfeReturnedImage enumeration members.
ParameterVersionHigh	Identifier specifying high part of library version static read-only parameter of type uint. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
ParameterVersionLow	Identifier specifying low part of library version static read-only parameter of type uint. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

7.13.1.1.1. VFExtractor Constructor

Initializes a new instance of the VFExtractor class.

```
public VFExtractor();
```

See Also

VFExtractor Class

7.13.1.1.2. IsRegistered Property

Gets a value indicating whether VFExtractor library has been registered.

```
public static bool IsRegistered {get}
```

Property value

`true` if VFExtractor library is registered; otherwise, `false`.

7.13.1.1.3. Mode Property

Gets or sets scanners mode.

```
public uint Mode {get, set}
```

Property value

One of the Mode constants.

See Also

Mode

7.13.1.1.4. ReturnedImage Property

Gets or sets the image type.

```
public VfeReturnedImage ReturnedImage {get, set};
```

Property value

One of the VfeReturnedImage value.

See Also

VfeReturnedImage

7.13.1.1.5. CopyParameters Method

Copies the parameters values from one VFExtractor object to another.

```
public static void CopyParameters(
        VFExtractor sourceExtractor,
        VFExtractor destinationExtractor
);
```

Parameters

sourceExtractor	Object of the `VFExtractor` class.
destinationExtractor	Object of the `VFExtractor` class.

See Also

Constants

7.13.1.1.6. Dispose Method

Releases the resources used by VFExtractor.

```
public void Dispose();
```

7.13.1.1.7. Extract Methods

Extracts features from a fingerprint image.

7.13.1.1.7.1. Extract(NGrayscaleImage image, NFPosition position, NFImpressionType impressionType, out bool isLowQuality)

Extracts finger features from NGrayscaleImage object.

```
public byte[] Extract(
        NGrayscaleImage image,
        NFPosition position,
        NFImpressionType impressionType,
        bool isLowQuality
        );
```

Parameters

image	The NGrayscaleImage image.
position	The finger NFPosition.
impressionType	The finger NFImpressionType.
isLowQuality	`true` if fingerprint image is low quality; otherwise, `false`.

Return Values

The array of packed NFRecord.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

Extract

7.13.1.1.7.2. Extract(ushort width, ushort height, uint stride, ushort horzResolution, ushort vertResolution, IntPtr image,NFPosition position, NFImpressionType impressionType, out bool isLowQuality)

Extracts finger features from fingerprint image.

```
public byte[] Extract(
        ushort width,
        ushort height,
        uint stride,
        ushort horzResolution,
        ushort vertResolution,
        IntPtr pixels,
        NFPosition position,
        NFImpressionType impressionType,
        out bool isLowQuality
        )
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
stride	The fingerprint image width, plus the width of the alignment bytes.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.
pixels	The pointer to array of fingerprint image.
position	The finger NFPosition.
impressionType	The finger NFImpressionType.
isLowQuality	`true` if fingerprint image is low quality; otherwise, `false`.

Return Values

The array of packed NFRecord.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

Extract

7.13.1.1.8. ExtractUnpacked Method

Extracts features from a fingerprint image and do not packs.

7.13.1.1.8.1. ExtractUnpacked(NGrayscaleImage,NFPosition,NFImpressionType,out bool)

```
public NFRecord ExtractUnpacked(
        NGrayscaleImage image,
        NFPosition position,
        NFImpressionType impressionType,
        out bool isLowQuality
);
```

Parameters

image	The NGrayscaleImage image.
position	The finger NFPosition.
impressionType	The finger NFImpressionType.
isLowQuality	`true` if fingerprint image is low quality; otherwise, `false`.

Return Values

The NFRecord object.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

NFRecord

7.13.1.1.8.2. ExtractUnpacked(ushort,ushort,uint,ushort,ushort,IntPtr,NFPosition,NFImpressionType,out bool)

```
public NFRecord ExtractUnpacked(
        ushort width,
        ushort height,
        uint stride,
        ushort horzResolution,
        ushort vertResolution,
```

```
        IntPtr pixels,
        NFPosition position,
        NFImpressionType impressionType,
        out bool isLowQuality
);
```

Parameters

width	The fingerprint image width.
height	The fingerprint image height.
stride	The fingerprint image width, plus the width of the alignment bytes.
horzResolution	Horizontal resolution of fingerprint image.
vertResolution	Vertical resolution of fingerprint image.
pixels	The pointer to array of fingerprint image.
position	The finger NFPosition.
impressionType	The finger NFImpressionType.
isLowQuality	`true` if fingerprint image is low quality; otherwise, `false`.

Return Values

The NFRecord object.

Remarks

position and *impressionType* are written to extracted NFRecord.

See Also

NFRecord

7.13.1.1.9. GetMaxTemplateSize Method

Retrieves maximal size of packed NFRecord the specified VFExtractor can extract.

```
public int GetMaxTemplateSize();
```

Return Values

Returns maximal template size.

7.13.1.1.10. GetParameter Method

Retrieves parameter by parameter identifier.

```
public object GetParameter(
        ushort parameterId
);
```

Parameters

parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterMode
- ParameterReturnedImage

See Also

SetParameter | CopyParameters

7.13.1.1.11. GetStaticParameter Method

Retrieves static parameter by parameter identifier.

```
public static object GetStaticParameter(
        ushort parameterId
);
```

Parameters

parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterCopyright
- ParameterName
- ParameterVersionHigh
- ParameterVersionLow

See Also

SetStaticParameter | CopyParameters

7.13.1.1.12. Register Method

Registers Neurotec.Biometrics.VFExtractor library.

```
public static void Register(
        string server
);
```

Parameters

server	The string that specifies license manager server name or IP address.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

7.13.1.1.13. Reset Method

Resets all parameters of VFExtractor object to default values.

```
public void Reset();
```

7.13.1.1.14. SetParameter Method

Sets the parameter by parameter identifier.

```
public void SetParameter(
        ushort parameterId,
        object value
);
```

Parameters

parameterId	The parameter identifier.
value	The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterMode
- ParameterReturnedImage

See Also

GetParameter | CopyParameters

7.13.1.1.15. SetStaticParameter Method

Sets the static parameter by parameter identifier.

```
public static void SetStaticParameter(
        ushort parameterId,
        object value
);
```

Parameters

parameterId	The parameter identifier.
value	The parameter object.

Remarks

Unused method.

See Also

GetStaticParameter | CopyParameters

7.13.1.1.16. Unregister Method

Unregisters Neurotec.Biometrics.VFExtractor library.

```
public static void Unregister();
```

See Also

Register | IsRegistered

7.13.1.2. VfeReturnedImage Enumeration

Specifies the returned image.

```
public enum VfeReturnedImage
```

Members

Member name	Description
None	No image.
Binarized	The binarized image.
Skeletonized	The skeletonized image.

7.14. Neurotec.Biometrics.VFMatcher Library

Provides functionality for comparing Neurotechnologija Finger Records using VeriFinger algorithm.

DLL: `Neurotec.Biometrics.VFMatcher.dll`.

Namespaces

Neurotec.Biometrics	Provides functionality for comparing Neurotechnologija Finger Records using VeriFinger algorithm.

7.14.1. Neurotec.Biometrics Namespace

Provides functionality for comparing Neurotechnologija Finger Records using VeriFinger algorithm.

Classes

VfmMatchDetails	Specifies the matching information.
VFMatcher	Provides functionality for comparing Neurotechnologija Finger Records (NFRecords) using VeriFinger algorithm encapsulated in

	Neurotechnologija Finger Matcher VF (VFMatcher) object.

Enumerations

VfmSpeed	Specifies the matching speed.

7.14.1.1. VfmMatchDetails Class

Specifies the matching information.

Properties

MatedMinutiae	The array of structure NIndexPair.
RawRotation	The rotation of fingerprint image.
Rotation	The rotation of fingerprint image in radians.
Score	Specifies score of two matched fingerprints.
TranslationX	The x translation of fingerprint image.
TranslationY	The y translation of fingerprint image.

Methods

Dispose	Releases the resources used by VfmMatch-Details.

7.14.1.1.1. MatedMinutiae Property

The array of structure NIndexPair.

```
public NIndexPair[] MatedMinutiae {get;}
```

Property value

The array of NIndexPair structures.

See Also

NIndexPair

7.14.1.1.2. RawRotation Property

The rotation of fingerprint image.

```
public byte RawRotation {get;}
```

7.14.1.1.3. Rotation Property

The rotation of fingerprint image in radians.

```
public double Rotation {get;}
```

7.14.1.1.4. Score Property

Specifies score of two matched fingerprints.

```
public int Score {get;}
```

Property value

The score of two matched fingerprints.

7.14.1.1.5. TranslationX Property

The x translation of fingerprint image.

```
public int TranslationX {get;}
```

7.14.1.1.6. TranslationY Property

The y translation of fingerprint image.

```
public int TranslationY {get;}
```

7.14.1.1.7. Dispose Method

Releases the resources used by VfmMatchDetails.

```
public void Dispose();
```

7.14.1.2. VFMatcher Class

Provides functionality for comparing Neurotechnologija Finger Records (NFRecords) using VeriFinger algorithm encapsulated in Neurotechnologija Finger Matcher VF (VFMatcher) object.

Constructors

VFMatcher Constructor	Initializes a new instance of the `VFMatcher` class.

Properties

IsRegistered	Gets a value indicating whether Neurotec.Biometrics.VFMatcher library is registered.
MatchingSpeed	Gets or sets matching speed.
MatchingThreshold	Gets or sets the matching threshold.
MaximalRotation	Gets or sets maximal rotation.
Mode	Gets or sets scanners mode.

Methods

CopyParameters	Copies parameters values from one VFMatcher object to another VFMatcher object.
Dispose	Releases the resources used by VFMatcher.
GetParameter	Gets value of parameter by parameter id.
GetStaticParameter	Gets value of static parameter by parameter id.
IdentifyEnd	Ends identification process.
IdentifyNext	Compares the specified NFRecord with one identification was started with.
IdentifyStart	Starts identification with the specified NFRecord.
Register	Registers Neurotec.Biometrics.VFMatcher library.
Reset	Resets all VFMatcher parameters to default values.
SetParameter	Sets value of parameter by parameter id.

Copyright © 2006 Neurotechnologija

SetStaticParameter	Sets value of static parameter by parameter id.
Unregister	Unregisters Neurotec.Biometrics.VFMatcher library.
Verify	Compars two NFRecords.

Constants

DllName	Name of DLL containing unmanaged part of this class.
ModeAtmelFingerchip ModeAuthentecAes4000 ModeAuthentecAFS2 ModeBiometrikaFX2000 ModeBmfBlp100 ModeCrossmatchVerifier300 ModeDigitalPersonaUareU ModeEthentica ModeGeneral ModeIdenticatorTechnologyDF90 ModeIdentixTouchView ModeKeytronicSecureDesktop ModePreciseBiometrics100CS ModeSecugenHamster ModeSTMicroelectronicsTouchChip	Represents scanners.
ParameterCopyright	Identifier specifying library copyright static read-only parameter of type string.
ParameterMatchingSpeed	Identifier specifying matching speed parameter of type int. Parameter value can be one of the VfmSpeed enumeration members.
ParameterMatchingThreshold	Identifier specifying matching threshold (biggest allowed FAR) parameter of type int. Parameter value is equal to - 12 * log10(FAR) and must be not less than zero (for example, 48 for FAR = 0.01%).
ParameterMaximalRotation	Identifier specifying modulus of maximal rotation allowed between two matched NFRecords parameter of type byte. Parameter value is specified in 180/128 degrees units and can not be greater than 128 (+-180 degrees).

Copyright © 2006 Neurotechnologija

ParameterMode	Identifier specifying mode (parameter value set) parameter of type uint. Parameter value can be one of the ModeXXX.
ParameterName	Identifier specifying library name static read-only parameter of type string.
ParameterVersionHigh	Identifier specifying high part of library version static read-only parameter of type uint. Two high-order bytes of parameter value specify major version and two low-order bytes - minor version.
ParameterVersionLow	Identifier specifying low part of library version static read-only parameter of type uint. Two high-order bytes of parameter value specify major (build) version and two low-order bytes - minor (release) version.

7.14.1.2.1. VFMatcher Constructor

Initializes a new instance of the VFMatcher class.

```
public VFMatcher();
```

7.14.1.2.2. IsRegistered Property

Gets a value indicating whether Neurotec.Biometrics.VFMatcher library is registered.

```
public static bool IsRegistered {get;}
```

Property value

`true` if Neurotec.Biometrics.VFMatcher library is registered; otherwise, `false`.

See Also

Register | Unregister

7.14.1.2.3. MatchingSpeed Property

Gets or sets matching speed.

```
public VfmSpeed MatchingSpeed {get; set;}
```

Property value

One of the VfmSpeed values. The default is `VfmSpeed.Low`.

See Also

VfmSpeed

7.14.1.2.4. MatchingThreshold Property

Gets or sets the matching threshold.

```
public int MatchingThreshold {get; set;}
```

Property value

The matching threshold.

See Also

ParameterMatchingThreshold

7.14.1.2.5. MaximalRotation Property

Gets or sets maximal rotation.

```
public byte MaximalRotation {get; set;}
```

Property value

The maximal rotation.

See Also

ParameterMaximalRotation

7.14.1.2.6. Mode Property

```
public uint Mode {get; set;}
```

Property value

The scanner type.

See Also

Constants

7.14.1.2.7. CopyParameters Method

Copies parameters values from one VFMatcher object to another VFMatcher object.

```
public static void CopyParameters(
        VFMatcher sourceMatcher,
        VFMatcher destinationMatcher
);
```

Parameters

sourceMatcher	The VFMatcher object.
destinationMatcher	The VFMatcher object.

See Also

Constants

7.14.1.2.8. Dispose Method

Releases the resources used by VFMatcher.

```
public void Dispose();
```

7.14.1.2.9. GetParameter Method

Gets value of parameter by parameter id.

```
public object GetParameter(
        ushort parameterId
);
```

Parameters

parameterId	The parameter identifier.

Return Values

The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterMatchingSpeed
- ParameterMatchingThreshold

Copyright © 2006 Neurotechnologija

- ParameterMaximalRotation
- ParameterMode

See Also

SetParameter | CopyParameters

7.14.1.2.10. GetStaticParameter Method

Gets value of static parameter by parameter id.

```
public static object GetStaticParameter(
        ushort parameterId
);
```

Parameters

parameterId	The parameter identifier.

Return Values

The parameter value.

Return Values

The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterCopyright
- ParameterName
- ParameterVersionHigh
- ParameterVersionLow

See Also

SetStaticParameter | CopyParameters

7.14.1.2.11. IdentifyEnd Method

Ends identification process.

```
public void IdentifyEnd();
```

Remarks

See Also

IdentifyStart | IdentifyNext

7.14.1.2.12. IdentifyNext Method

Compares the specified NFRecord with one identification was started with.

7.14.1.2.12.1. IdentifyNext(IntPtr,int)

```
public int IdentifyNext(
        IntPtr template,
        int templateSize
);
```

Parameters

template	Pointer to the memory block that contains packed NFRecord.
templateSize	The template size.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NFRecord

7.14.1.2.12.2. IdentifyNext(IntPtr,int,VfmMatchDetails)

```
public int IdentifyNext(
        IntPtr template,
        int templateSize,
        VfmMatchDetails matchDetails
);
```

Parameters

template	Pointer to the memory block that contains packed NFRecord.
templateSize	The template size.
matchDetails	The VfmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | VfmMatchDetails | NFRecord

7.14.1.2.12.3. IdentifyNext(byte[])

```
public int IdentifyNext(
        byte[] template
);
```

Parameters

template	The byte array with packed NFRecord.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | NFRecord

Copyright © 2006 Neurotechnologija

7.14.1.2.12.4. IdentifyNext(byte[],VfmMatchDetails)

```
public int IdentifyNext(
        byte[] template,
        VfmMatchDetails matchDetails
);
```

Parameters

template	The byte array with packed NFRecord.
matchDetails	The VfmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

IdentifyStart | IdentifyEnd | VfmMatchDetails | NFRecord

7.14.1.2.13. IdentifyStart Method

Starts identification with the specified NFRecord.

7.14.1.2.13.1. IdentifyStart(IntPtr template, int templateSize)

```
public void IdentifyStart(
        IntPtr template,
        int templateSize
);
```

Parameters

template	Pointer to the memory block that contains packed NFRecord.
templateSize	The template size.

See Also

IdentifyNext | IdentifyEnd | NFRecord

7.14.1.2.13.2. IdentifyStart(IntPtr,int,out VfmMatchDetails)

```
public void IdentifyStart(
        IntPtr template,
        int templateSize,
        out VfmMatchDetails matchDetails
);
```

Parameters

template	Pointer to the memory block that contains packed NFRecord.
templateSize	The template size.
matchDetails	The VfmMatchDetails object.

See Also

IdentifyNext | IdentifyEnd | VfmMatchDetails | NFRecord

7.14.1.2.13.3. IdentifyStart(byte[])

```
public void IdentifyStart(
        byte[] template
);
```

Parameters

template	The byte array with packed NFRecord.

See Also

IdentifyNext | IdentifyEnd | NFRecord

7.14.1.2.13.4. IdentifyStart(byte[],out VfmMatchDetails)

```
public void IdentifyStart(
        byte[] template,
        out VfmMatchDetails matchDetails
```

```
);
```

Parameters

template	The byte array with packed NFRecord.
matchDetails	The VfmMatchDetails object.

See Also

IdentifyNext | IdentifyEnd | VfmMatchDetails | NFRecord

7.14.1.2.14. Register Method

Registers Neurotec.Biometrics.VFMatcher library.

```
public static void Register(
        string server
);
```

Parameters

server	The string that specifies license manager server name or IP address.

Remarks

When library is no longer needed (possibly during application shutdown) it must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

7.14.1.2.15. Reset Method

Resets all VFMatcher parameters to default values.

```
public void Reset();
```

7.14.1.2.16. SetParameter Method

Sets value of parameter by parameter id.

```
public void SetParameter(
        ushort parameterId,
```

```
        object value
);
```

Parameters

parameterId	The parameter identifier.
value	The parameter value.

Remarks

The following values can be used for *parameterId*:

- ParameterMatchingSpeed
- ParameterMatchingThreshold
- ParameterMaximalRotation
- ParameterMode

See Also

GetParameter | CopyParameters

7.14.1.2.17. SetStaticParameter Method

Sets value of static parameter by parameter id.

```
public static void SetStaticParameter(
        ushort parameterId,
        object value
);
```

Parameters

parameterId	The parameter identifier.
value	The parameter value.

Remarks

Unused mothod.

See Also

GetStaticParameter | CopyParameters

7.14.1.2.18. Unregister Method

Unregisters Neurotec.Biometrics.VFMatcher library.

```
public static void Unregister();
```

See Also

Register | IsRegistered

7.14.1.2.19. Verify Method

Compars two NFRecords.

7.14.1.2.19.1. Verify(IntPtr,int,IntPtr,int)

```
public int Verify(
        IntPtr template1,
        int template1Size,
        IntPtr template2,
        int template2Size
);
```

Parameters

template1	Pointer to the memory block that contains packed NFRecord.
template1Size	The template size.
template2	Pointer to the memory block that contains packed NFRecord.
template2Size	The template size.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NFRecord

7.14.1.2.19.2. Verify(IntPtr,int,IntPtr,int,out VfmMatchDetails)

```
public int Verify(
        IntPtr template1,
        int template1Size,
        IntPtr template2,
        int template2Size,
        out VfmMatchDetails matchDetails
);
```

Parameters

template1	Pointer to the memory block that contains packed NFRecord.
template1Size	The template size.
template2	Pointer to the memory block that contains packed NFRecord.
template2Size	The template size.
matchDetails	The VfmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

VfmMatchDetails | NFRecord

7.14.1.2.19.3. Verify(byte[],byte[])

```
public int Verify(
        byte[] template1,
        byte[] template2
);
```

Parameters

template1	The byte array with packed NFRecord.
template2	The byte array with packed NFRecord.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

NFRecord

7.14.1.2.19.4. Verify(byte[],byte[],out VfmMatchDetails)

```
public int Verify(
        byte[] template1,
        byte[] template2,
        out VfmMatchDetails matchDetails
);
```

Parameters

template1	The byte array with packed NFRecord.
template2	The byte array with packed NFRecord.
matchDetails	The VfmMatchDetails object.

Return Values

The matching score.

Remarks

Method returns zero if similarity is less than matching threshold, i.e. two NFRecords do not match (see ParameterMatchingThreshold and SetParameter method), and is greater than or equal to matching threshold otherwise.

See Also

VfmMatchDetails | NFRecord

7.14.1.3. VfmSpeed Enumeration

Specifies the matching speed.

```
public enum VfmSpeed
```

Members

Member	Description
High	The high matching speed.
Low	The low matching speed.

7.15. Neurotec.Images Pro Library

Provides classes that enable loading, saving and converting images in various formats.

DLL: `Neurotec.Images.dll`.

Namespaces

Neurotec.Images	Contains classes that enable loading, saving and converting images in various formats.

7.15.1. Neurotec.Images Namespace

Contains classes that enable loading, saving and converting images in various formats.

Classes

Bmp	Provides functionality for loading and saving images in BMP format.
IHead	Provides functionality for loading and saving images in NIST IHead format.
NGrayscaleImage	Provides functionality for managing 8-bit grayscale images.
NImage	Provides functionality for managing images.

NImageFile	Provides functionality for reading image files in format-neutral style.
NImageFormat	Provides functionality for loading and saving images in format-neutral style.
NImageFormat.ImageFormatCollection	Represents the collection of formats in a NImageFormat.
NImages	Provides library registration and other additional functionality.
NMonochromeImage	Provides functionality for managing 1-bit monochrome images.
NRgbImage	Provides functionality for managing 24-bit RGB images.
Tiff	Provides functionality for loading and saving images in TIFF format.
Wsq	Provides functionality for loading and saving images in WSQ format.

Structures

NPixelFormat	Provides functionality for work with pixel format.
NRgb	Represents an RGB color.

7.15.1.1. Bmp Class

Provides functionality for loading and saving images in BMP format.

Methods

LoadImage	Creates a new instance of the `NImage` class.
LoadImageFromBitmap	Creates a new instance of the `NImage` class from `Bitmap`.
LoadImageFromHBitmap	Creates a new instance of the `NImage` class from Windows HBITMAP.
SaveImage	Saves `NImage`.

SaveImageToBitmap	Saves `NImage` to `Bitmap`.
SaveImageToHBitmap	Saves `NImage` to Windows HBITMAP.

7.15.1.1.1. LoadImage Method

Creates a new instance of the `NImage` class

7.15.1.1.1.1. LoadImage (string)

Creates a new instance of the `NImage` class from file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

7.15.1.1.1.2. LoadImage (IntPtr, int)

Creates a new instance of the `NImage` class from memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

7.15.1.1.1.3. LoadImage (byte[])

Creates a new instance of the `NImage` class from byte array.

```
public static NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

7.15.1.1.2. LoadImageFromBitmap Method

Creates a new instance of the `NImage` class from `Bitmap`.

7.15.1.1.2.1. LoadImageFromBitmap (Bitmap)

Creates a new instance of the `NImage` class from `Bitmap`.

```
public static NImage LoadImageFromBitmap(
        Bitmap bitmap
);
```

Parameters

bitmap	A `Bitmap` class object.

Return Values

A NImage object.

See Also

NImage | SaveImageToBitmap

7.15.1.1.2.2. LoadImageFromBitmap (Bitmap, float, float)

Creates a new instance of the `NImage` class from `Bitmap` with specified resolution.

```
public static NImage LoadImageFromBitmap(
        Bitmap bitmap,
        float horzResolution,
        float vertResolution
);
```

Parameters

bitmap	A `Bitmap` class object.
horzResolution	A horizontal resolution of fingerprint image.
vertResolution	A vertical resolution of fingerprint image.

Return Values

A NImage object.

See Also

NImage | SaveImageToBitmap

7.15.1.1.3. LoadImageFromHBitmap Method

Creates a new instance of the `NImage` class from Windows HBITMAP.

```
public static NImage LoadImageFromHBitmap(
        IntPtr hBitmap
);
```

Parameters

hBitmap	Pointer to handle that specifies Windows HBITMAP.

Return Values

A NImage object.

Copyright © 2006 Neurotechnologija

See Also

NImage | SaveImageToHBitmap

7.15.1.1.4. SaveImage Method

Saves `NImage`.

7.15.1.1.4.1. void SaveImage (NImage, string)

Saves `NImage` to file.

```
public static void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

NImage | LoadImage

7.15.1.1.4.2. void SaveImage (NImage, Stream)

Saves `NImage` to stream.

```
public static void SaveImage(
        NImage image,
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

NImage | LoadImage

7.15.1.1.4.3. byte[] SaveImage (NImage)

Saves `NImage` to byte array.

```
public static byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A byte array.

See Also

NImage | LoadImage

7.15.1.1.5. SaveImageToBitmap Method

Saves `NImage` to `Bitmap`.

```
public static Bitmap SaveImageToBitmap(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A `Bitmap` object.

See Also

NImage | LoadImageFromBitmap

7.15.1.1.6. SaveImageToHBitmap Method

Saves `NImage` to Windows HBITMAP.

```
public static IntPtr SaveImageToHBitmap(
        NImage image
```

```
);
```

Parameters

image	A NImage object.

Return Values

A pointer to handle of Windows HBITMAP.

See Also

NImage | LoadImageFromHBitmap

7.15.1.2. IHead Class

Provides functionality for loading and saving images in NIST IHead format.

Methods

LoadImage	Creates `NImage` object from NIST IHead.
SaveImage	Saves `NImage` object in NIST IHead format.

7.15.1.2.1. LoadImage Method

Creates `NImage` object from NIST IHead.

7.15.1.2.1.1. LoadImage (string)

Creates `NImage` object from NIST IHead file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage

7.15.1.2.1.2. LoadImage (IntPtr, int)

Creates `NImage` object from NIST IHead memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage

7.15.1.2.1.3. LoadImage (byte[])

Creates `NImage` object from NIST IHead byte array.

```
public static NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage

7.15.1.2.2. SaveImage Method

Saves `NImage` object in NIST IHead format.

7.15.1.2.2.1. void SaveImage (NImage, string)

Saves `NImage` object to file in NIST IHead format.

```
public static void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

NImage

7.15.1.2.2.2. byte[] SaveImage (NImage)

Saves `NImage` object to byte array in NIST IHead format.

```
public static byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A array with saved image.

See Also

NImage

7.15.1.2.2.3. void SaveImage (NImage, Stream)

Saves `NImage` object to memory stream in NIST IHead format.

```
public static void SaveImage(
        NImage image,
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

NImage

7.15.1.3. NGrayscaleImage Class

Provides functionality for managing 8-bit grayscale images.

Properties

Item	Gets or sets the color of the specified pixel in NImage object.

7.15.1.3.1. NGrayscaleImage.Item Property

Gets or sets the color of the specified pixel in NImage object.

```
public const byte this[
        uint x,
        uint y
] {get; set;}
```

Parameters

x	The x coordinate of the pixel.
y	The y coordinate of the pixel.

Property value

A color of specified pixel.

7.15.1.4. NImage Class

Provides functionality for managing images.

Properties

Handle	Gets handle to unmanaged NImage object.
Height	Gets height of fingerprint image from NImage object.
HorzResolution	Gets horizontal resolution of fingerprint image.
LongSize	Gets size of NImage object.
LongStride	Gets stride of fingerprint image from NImage object.
PixelFormat	Gets NPixelFormat of NImage object.
Pixels	Gets pointer to array of pixels from NImage object.
Size	Gets size of NImage object.
Stride	Gets stride of fingerprint image from NImage object.
VertResolution	Gets vertical resolution of fingerprint image.
Width	Gets width of fingerprint image from NImage object.

Methods

Clone	Creates NImage object from another NImage object.
Create	Creates an empty NImage object.
Dispose	Releases the resources used by NImage.
FromBitmap	Creates `NImage` from Bitmap.
FromData	Creates `NImage` object from data.
FromFile	Creates `NImage` object from file.
FromHandle	Creates `NImage` object from handle.
FromHBitmap	Creates a new instance of the NImage class from Windows HBITMAP.

Copyright © 2006 Neurotechnologija

FromImage	Creates `NImage` object from another `NImage` object.
GetWrapper	Creates `NImage` object wrapper.
Save	Saves `NImage` object to file.
ToBitmap	Creates a `Bitmap`.
ToHBitmap	Creates Windows HBITMAP.

7.15.1.4.1. Handle Property

Gets handle to unmanaged NImage object.

```
public IntPtr Handle {get;}
```

Property value

A handle to unmanaged NImage object.

See Also

NImage

7.15.1.4.2. Height Property

Gets height of fingerprint image from NImage object.

```
public uint Height {get;}
```

Property value

A height of fingerprint image.

See Also

NImage Width | Stride

7.15.1.4.3. HorzResolution Property

Gets horizontal resolution of fingerprint image.

```
public float HorzResolution {get;}
```

Property value

A horizontal resolution of fingerprint image.

Remarks

Horizontal resolution equal to zero means that it is not applicable for the image.

See Also

NImage VertResolution

7.15.1.4.4. LongSize Property

Gets size of NImage object.

```
public ulong LongSize {get;}
```

Property value

A size of NImage object.

Remarks

Size of memory block containing image pixels is equal to image height multiplied by image stride. For more information see Height and Stride properties.

See Also

NImage | Height | Stride

7.15.1.4.5. LongStride Property

Gets stride of fingerprint image from NImage object.

```
public ulong LongStride {get;}
```

Property value

A stride of fingerprint image.

Stride (size of one row) of the image depends on image pixel format and width. It can not be less than value obtained with GetRowLongSize or GetRowSize methods with arguments obtained with PixelFormat and Width properties.

See Also

NImage

7.15.1.4.6. PixelFormat Property

Gets NPixelFormat of NImage object.

```
public NPixelFormat PixelFormat {get;}
```

Property value

A NPixelFormat structure.

See Also

NImage

7.15.1.4.7. Pixels Property

Gets pointer to array of pixels from NImage object.

```
public IntPtr Pixels {get;}
```

Property value

A pointer to pixel array.

Remarks

Memory block containing image pixels is organized as image height rows following each other in top-to-bottom order. Each row occupies image stride bytes and is organized as image width pixels following each other in right-to-left order. Each pixel is described by image pixel format.

For more information see PixelFormat, Width, Height, Stride, and Size properties.

See Also

NImage | PixelFormat | Width | Height | Stride | Size

7.15.1.4.8. Size Property

Gets size of NImage object.

```
public uint Size {get;}
```

Property value

A size of NImage object.

Remarks

Size of memory block containing image pixels is equal to image height multiplied by image

stride. For more information see Height and Stride properties.

See Also

NImage | Height | Stride

7.15.1.4.9. Stride Property

Gets stride of fingerprint image from NImage object.

```
public uint Stride {get;}
```

Property value

A stride of fingerprint image.

Stride (size of one row) of the image depends on image pixel format and width. It can not be less than value obtained with GetRowLongSize or GetRowSize methods with arguments obtained with PixelFormat and Width properties.

See Also

NImage

7.15.1.4.10. VertResolution Property

Gets vertical resolution of fingerprint image.

```
public float VertResolution {get;}
```

Property value

A vertical resolution of fingerprint image.

Remarks

Vertical resolution equal to zero means that it is not applicable for the image.

See Also

NImage | HorzResolution

7.15.1.4.11. Width Property

Gets width of fingerprint image from NImage object.

```
public uint Width {get;}
```

Property value

A width of fingerprint image.

See Also

NImage Height | Stride

7.15.1.4.12. Clone Method

Creates NImage object from another NImage object.

```
public object Clone();
```

Return Values

A NImage object.

See Also

NImage

7.15.1.4.13. Create Method

Creates an empty NImage object.

7.15.1.4.13.1. Create (NPixelFormat, uint, uint, uint, float, float)

Creates an empty NImage object.

```
public static NImage Create(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        uint stride,
        float horzResolution,
        float vertResolution
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution of fingerprint image.

Copyright © 2006 Neurotechnologija

vertResolution	A vertical resolution of fingerprint image.

Return Values

A NImage object.

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride method.

horzResolutionM and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

NImage

7.15.1.4.13.2. Create (NPixelFormat, uint, uint, ulong, float, float)

Creates an empty NImage object.

```
public static NImage Create(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        ulong stride,
        float horzResolution,
        float vertResolution
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution of fingerprint image.
vertResolution	A vertical resolution of fingerprint image.

Return Values

A NImage object.

If *stride* is zero then image stride is automatically calculated. For more information on im-

age stride see Stride method.

horzResolutionM and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

NImage

7.15.1.4.14. Dispose Method

Releases the resources used by NImage.

```
public void Dispose();
```

See Also

NImage

7.15.1.4.15. FromBitmap Method

Creates `NImage` from Bitmap.

```
public static NImage FromBitmap(
        Bitmap bitmap
);
```

Parameters

bitmap	An object used to work with images defined by pixel data.

Return Values

A NImage object.

See Also

NImage

7.15.1.4.16. FromData Method

Creates `NImage` object from data.

7.15.1.4.16.1. FromData (NPixelFormat, uint, uint, uint, float, float, uint, IntPtr)

Creates `NImage` object from data with specified resolution.

```
public static NImage FromData(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        uint stride,
        float horzResolution,
        float vertResolution,
        uint srcStride,
        IntPtr srcPixels
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution of fingerprint image.
vertResolution	A vertical resolution of fingerprint image.
srcStride	A stride of source fingerprint image.
srcPixels	A pointer to source pixel array.

Return Values

A NImage object.

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride property.

Format of memory block *srcPixels* points to must be the same as described in Pixels property, only stride is equal to *srcStride*.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

NImage

7.15.1.4.16.2. FromData (NPixelFormat, uint, uint, ulong, float, float, ulong, IntPtr)

Creates `NImage` object from data with specified resolution.

Copyright © 2006 Neurotechnologija

```
public static NImage FromData(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        ulong stride,
        float horzResolution,
        float vertResolution,
        ulong srcStride,
        IntPtr srcPixels
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution of fingerprint image.
vertResolution	A vertical resolution of fingerprint image.
srcStride	A stride of source fingerprint image.
srcPixels	A pointer to source pixel array.

Return Values

A NImage object.

See Also

NImage

7.15.1.4.17. FromFile Method

Creates `NImage` object from file.

7.15.1.4.17.1. FromFile (string)

Creates `NImage` object from file.

```
public static NImage FromFile(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage

7.15.1.4.17.2. FromFile (string, NImageFormat)

Creates `NImage` object from file with specified NImageFormat.

```
public static NImage FromFile(
        string fileName,
        NPixelFormat imageFormat
);
```

Parameters

fileName	A string that contains the name of the file.
imageFormat	An image NImageFormat object.

Return Values

A NImage object.

See Also

NImage | NImageFormat

7.15.1.4.18. FromHandle Method

Creates `NImage` object from handle.

```
public static NImage FromHandle(
        IntPtr handle
);
```

Parameters

handle	A pointer to handle.

Return Values

A NImage object.

See Also

NImage | Handle

7.15.1.4.19. FromHBitmap Method

Creates a new instance of the NImage class from Windows HBITMAP.

```
public static NImage FromHBitmap(
        IntPtr hBitmap
);
```

Parameters

hBitmap	Pointer to handle that specifies Windows HBITMAP.

Return Values

A NImage object.

See Also

NImage

7.15.1.4.20. FromImage Method

Creates `NImage` object from another `NImage` object.

7.15.1.4.20.1. FromImage (NPixelFormat, uint, NImage)

Creates `NImage` object from another `NImage` object.

```
public static NImage FromImage(
        NPixelFormat pixelFormat,
        uint stride,
        NImage srcImage
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
stride	A stride of fingerprint image.

srcImage	A `NImage` source object.

Return Values

A NImage object.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride property.

See Also

NImage | NPixelFormat

7.15.1.4.20.2. FromImage (NPixelFormat, ulong, NImage)

Creates `NImage` object from another `NImage` object.

```
public static NImage FromImage(
        NPixelFormat pixelFormat,
        ulong stride,
        NImage srcImage
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
stride	A stride of fingerprint image.
srcImage	A `NImage` source object.

Return Values

A NImage object.

See Also

NImage | NPixelFormat

7.15.1.4.20.3. FromImage (NPixelFormat, uint, float, float, NImage)

Creates `NImage` object from another `NImage` object with specified resolution.

```
public static NImage FromImage(
        NPixelFormat pixelFormat,
```

```
        uint stride,
        float horzResolution,
        float vertResolution,
        NImage srcImage
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution of fingerprint image.
vertResolution	A vertical resolution of fingerprint image.
srcImage	A `NImage` source object.

Return Values

A NImage object.

Remarks

If *stride* is zero then image stride is automatically calculated. For more information on image stride see Stride property.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

NImage | NPixelFormat

7.15.1.4.20.4. FromImage (NPixelFormat, ulong, float, float, NImage)

Creates `NImage` object from another `NImage` object with specified resolution.

```
public static NImage FromImage(
        NPixelFormat pixelFormat,
        ulong stride,
        float horzResolution,
        float vertResolution,
        NImage srcImage
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.

stride	A stride of fingerprint image.
horzResolution	A horizontal resolution of fingerprint image.
vertResolution	A vertical resolution of fingerprint image.
srcImage	A `NImage` source object.

Return Values

A NImage object.

See Also

NImage | NPixelFormat

7.15.1.4.21. GetWrapper Method

Creates `NImage` object wrapper.

7.15.1.4.21.1. GetWrapper (NPixelFormat, uint, uint, uint, float, float, IntPtr, bool)

Creates `NImage` object wrapper.

```
public static NImage GetWrapper(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        uint stride,
        float horzResolution,
        float vertResolution,
        IntPtr pixels,
        bool ownsPixels
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.
stride	A stride of fingerprint image.
horzResolution	A horizontal resolution of fingerprint image.
vertResolution	A vertical resolution of fingerprint image.

Copyright © 2006 Neurotechnologija

pixels	Pointer to memory block containing pixels for the image.
ownsPixels	Specifies whether pixels will be automatically deleted with the image (if set to true).

Return Values

A NImage object.

Remarks

For more information on image stride see Stride property.

Format of memory block pixels points to must be the same as described in Pixels property.

pixels must not be deleted during lifetime of the image. If *ownsPixels* is `true` then pixels will be automatically deleted with the image.

horzResolution and *vertResolution* can be zero if resolution is not applicable for the image.

See Also

NImage | NPixelFormat | Pixels

7.15.1.4.21.2. GetWrapper (NPixelFormat, uint, uint, ulong, float, float, IntPtr, bool)

Creates `NImage` object wrapper.

```
public static NImage GetWrapper(
        NPixelFormat pixelFormat,
        uint width,
        uint height,
        ulong stride,
        float horzResolution,
        float vertResolution,
        IntPtr pixels,
        bool ownsPixels
);
```

Parameters

pixelFormat	A NPixelFormat of fingerprint image.
width	A width of fingerprint image.
height	A height of fingerprint image.

stride	A stride of fingerprint image.
horzResolution	A horizontal resolution of fingerprint image.
vertResolution	A vertical resolution of fingerprint image.
pixels	Pointer to memory block containing pixels for the image.
ownsPixels	Specifies whether pixels will be automatically deleted with the image (if set to true).

Return Values

A NImage object.

Remarks

Format of memory block pixels points to must be the same as described in Pixels property.

See Also

NImage | NPixelFormat | Pixels

7.15.1.4.22. Save Method

Saves `NImage` object to file.

7.15.1.4.22.1. Save (string)

Saves `NImage` object to file.

```
public void Save(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

See Also

NImage

7.15.1.4.22.2. Save (string, NImageFormat)

Saves `NImage` object to file with specified NImageFormat.

```
public void Save(
        string fileName,
        NImageFormat imageFormat
);
```

Parameters

fileName	A string that contains the name of the file.
imageFormat	An image NImageFormat object.

See Also

NImage | NImageFormat

7.15.1.4.23. ToBitmap Method

Creates a `Bitmap`.

```
public Bitmap ToBitmap();
```

Return Values

A `Bitmap` object.

See Also

NImage

7.15.1.4.24. ToHBitmap Method

Creates Windows HBITMAP.

```
public IntPtr ToHBitmap();
```

Return Values

A Windows HBITMAP.

See Also

NImage

7.15.1.5. NImageFile Class

Provides functionality for reading image files in format-neutral style.

Properties

Format	Gets image format from `NImageFile` object.
IsOpened	Gets a value indicating whether the file is currently open.

Methods

Close	Closes a file which is associated with `NImageFile` object.
Dispose	Releases the resources used by `NImageFile`.
FromFile	Creates `NImageFile` object from file.
ReadImage	Reads image from `NImageFile` object.

7.15.1.5.1. Format Property

Gets image format from `NImageFile` object.

```
public virtual NImageFormat Format {get;}
```

Property value

A NImageFormat object.

7.15.1.5.2. IsOpened Property

Gets a value indicating whether the file is currently open.

```
public virtual bool IsOpened {get;}
```

Property value

`true` if file is open, `false` if file is closed.

See Also

Close

7.15.1.5.3. Close Method

Closes a file which is associated with `NImageFile` object.

```
public virtual void Close();
```

See Also

IsOpened

7.15.1.5.4. Dispose Method

Releases the resources used by `NImageFile`.

```
public void Dispose();
```

7.15.1.5.5. FromFile Method

Creates `NImageFile` object from file.

7.15.1.5.5.1. FromFile (string)

Creates `NImageFile` object from file.

```
public static NImageFile FromFile(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImageFile

See Also

A NImageFile

7.15.1.5.5.2. FromFile (string, NImageFormat)

Creates `NImageFile` object from file with specified image format.

```
public static NImageFile FromFile(
        string fileName,
        NImageFormat imageFormat
);
```

Parameters

fileName	A string that contains the name of the file.
imageFormat	An image NImageFormat object.

Return Values

A NImageFile

See Also

NImageFile | NImageFormat

7.15.1.5.6. ReadImage Method

Reads image from `NImageFile` object.

```
public virtual NImage ReadImage();
```

Return Values

A NImage object.

See Also

NImage

7.15.1.6. NImageFormat Class

Provides functionality for loading and saving images in format-neutral style.

Fields

Bmp	Specifies the BMP image format.
Formats	Specifies collection of supported image formats.
Gif	Specifies the GIF image format.
IHead	Specifies the NIST IHead image format.
Jpeg	Specifies the JPEG image format.
Png	Specifies the PNG image format.
Tiff	Specifies the TIFF image format.

Copyright © 2006 Neurotechnologija

Wsq	Specifies the WSQ image format.

Properties

CanRead	Gets a value indicating whether the current image format supports reading.
CanWrite	Gets a value indicating whether the current image format supports writing.
CanWriteMultiple	Gets a value indicating whether the current image format supports writing of multiple images.
DefaultFileExtension	Gets default file extension of the current image format.
FileFilter	Gets file filter of the current image format.
Name	Gets name of the current image format.

Methods

LoadImage	Loads `NImage`.
OpenFile	Opens `NImageFile`.
SaveImage	Saves `NImage`.
SaveImages	Saves array of `NImage`.
Select	Returns supported image format registered with file extension of specified file name and supporting reading/writing as specified.

7.15.1.6.1. Bmp Field

```
public static readonly NImageFormat Bmp;
```

7.15.1.6.2. Formats Field

```
public static readonly NImageFormat.ImageFormatCollection Formats;
```

7.15.1.6.3. Gif Field

```
public static readonly NImageFormat Gif;
```

7.15.1.6.4. IHead Field

```
public static readonly NImageFormat IHead;
```

7.15.1.6.5. Jpeg Field

```
public static readonly NImageFormat Jpeg;
```

7.15.1.6.6. Png Field

```
public static readonly NImageFormat Png;
```

7.15.1.6.7. Tiff Field

```
public static readonly NImageFormat Tiff;
```

7.15.1.6.8. Wsq Field

```
public static readonly NImageFormat Wsq;
```

7.15.1.6.9. CanRead Property

```
public virtual bool CanRead {get;}
```

Property value

`true` if image format can read, `false` if image format can not read.

See Also

CanWrite | CanWriteMultiple | Name | DefaultFileExtension | FileFilter

7.15.1.6.10. CanWrite Property

```
public virtual bool CanWrite {get;}
```

Property value

`true` if image format can write, `false` if image format can not write.

See Also

CanRead | CanWriteMultiple | Name | DefaultFileExtension | FileFilter

7.15.1.6.11. CanWriteMultiple Property

```
public virtual bool CanWriteMultiple {get;}
```

Property value

`true` if image format can write the multiple images, `false` if image format can not write the multiple images.

See Also

CanRead | CanWrite | Name | DefaultFileExtension | FileFilter

7.15.1.6.12. DefaultFileExtension Property

```
public virtual string DefaultFileExtension {get;}
```

Property value

Default file extension.

See Also

CanRead | CanWrite | Name | DefaultFileExtension | FileFilter

7.15.1.6.13. FileFilter Property

```
public virtual string FileFilter {get;}
```

Property value

An image format file filter

See Also

CanRead | CanWrite | Name | DefaultFileExtension | CanWriteMultiple

7.15.1.6.14. Name Property

```
public virtual string Name {get;}
```

Property value

An image format name.

Copyright © 2006 Neurotechnologija

See Also

CanRead | CanWrite | FileFilter | DefaultFileExtension | CanWriteMultiple

7.15.1.6.15. LoadImage Method

Loads `NImage`.

7.15.1.6.15.1. LoadImage(string)

Loads `NImage` from file.

```
public virtual NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | SaveImage

7.15.1.6.15.2. LoadImage(IntPtr, int)

Loads `NImage` from memory buffer.

```
public virtual NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

Copyright © 2006 Neurotechnologija

See Also

NImage | SaveImage

7.15.1.6.15.3. LoadImage(byte[])

Loads `NImage` from byte array.

```
public virtual NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | SaveImage

7.15.1.6.16. OpenFile Method

Opens `NImageFile`.

7.15.1.6.16.1. OpenFile(string)

Opens `NImageFile`.

```
public virtual NImageFile OpenFile(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImageFile object.

See Also

NImageFile | OpenFile

7.15.1.6.16.2. OpenFile(IntPtr, int)

Opens `NImageFile` from memory buffer.

```
public virtual NImageFile OpenFile(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImageFile object.

See Also

NImageFile | OpenFile

7.15.1.6.16.3. OpenFile(byte[])

Opens `NImageFile` from byte array.

```
public virtual NImageFile OpenFile(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImageFile object.

See Also

NImageFile | OpenFile

7.15.1.6.17. SaveImage Method

Saves NImage.

7.15.1.6.17.1. void SaveImage (NImage, string)

Saves NImage to file.

```
public virtual void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

LoadImage | NImage

7.15.1.6.17.2. byte[] SaveImage(NImage)

Saves NImage to byte array.

```
public virtual byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A byte array.

See Also

LoadImage | NImage

7.15.1.6.17.3. void SaveImage(NImage, Stream)

Saves NImage to stream.

```
public virtual void SaveImage(
        NImage image,
```

```
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

LoadImage | NImage

7.15.1.6.18. SaveImages Method

Saves array of NImage.

7.15.1.6.18.1. void SaveImages (NImage[], string)

Saves array of NImage to file.

```
public virtual void SaveImages(
        NImage[] images,
        string fileName
);
```

Parameters

images	A NImage objects array.
fileName	A string that contains the name of the file.

See Also

LoadImage | NImage

7.15.1.6.18.2. byte[] SaveImages(NImage[])

Saves array of NImage to byte array.

```
public virtual byte[] SaveImages(
        NImage[] images
);
```

Parameters

images	A NImage objects array.

Return Values

A byte array.

See Also

LoadImage | NImage

7.15.1.6.18.3. void SaveImages(NImage[], Stream)

Saves array of NImage to stream.

```
public virtual void SaveImages(
        NImage[] images,
        Stream stream
);
```

Parameters

images	A NImage objects array.
stream	The data stream used to save the image.

See Also

LoadImage | NImage

7.15.1.6.19. Select Method

Retrieves supported image format registered with file extension of specified file name and supporting reading/writing as specified.

```
public static NImageFormat Select(
        string fileName,
        FileAccess fileAccess
);
```

Parameters

fileName	A string that contains the name of the file.
fileAccess	A file access.

Return Values

A NImageFormat object.

See Also

NImageFormat

7.15.1.7. NImageFormat.ImageFormatCollection Class

Represents the collection of formats in a NImageFormat.

Properties

Item	Gets the member from collection by index.

Methods

IndexOf	Returns the index within the collection of the specified image format.

7.15.1.7.1. ImageFormatCollection.Item Property

Gets the member from collection by index.

```
public NImageFormat this[
        int index
] {get;}
```

Parameters

x	The index of the element to get.

Property value

A NImageFormat object.

See Also

NImageFormat

7.15.1.7.2. IndexOf Method

Returns the index within the collection of the specified image format.

```
public int IndexOf(
        NImageFormat value
);
```

Parameters

value	A NImageFormat object.

Return Values

The zero-based index of the NImageFormat in the collection.

See Also

NImageFormat

7.15.1.8. NImages Class

Provides library registration and other additional functionality.

Properties

IsRegistered	Checks if Neurotec.NImages Pro library is registered.

Methods

GetGrayscaleColorWrapper	Creates NImage object wrapper.
Register	Registers Neurotec.NImages Pro library.
Unregister	Unregisters Neurotec.NImages Pro library.

Constants

DllName	Name of DLL containing unmanaged part of this class.

7.15.1.8.1. IsRegistered Property

Checks if Neurotec.NImages Pro library is registered.

```
public static bool IsRegistered {get;}
```

Copyright © 2006 Neurotechnologija

Property value

`true` if library is registered, `false` if library is not registered.

See Also

Register | Unregister

7.15.1.8.2. GetGrayscaleColorWrapper Method

Creates NImage object wrapper.

```
public static NImage GetGrayscaleColorWrapper(
        NImage image,
        NRgb minColor,
        NRgb maxColor
);
```

Parameters

image	A NImage object.
minColor	Specifies color to be used for black color.
maxColor	Specifies color to be used for white color.

Return Values

An NImage object.

Remarks

Gray values in source image are replaced with according RGB values from range [*minColor*, *maxColor*] in created image.

See Also

NImage

7.15.1.8.3. Register Method

Registers Neurotec.NImages Pro library.

```
public static void Register(
        string server
);
```

Parameters

Copyright © 2006 Neurotechnologija

server	The string that specifies license manager server name or IP address (if registering with LAN License Manager), any string (if registering with HASP) or serial number and registration key separated by '@' (if registering using PC or U.are.U protection).

Remarks

If registering with HASP it is recommended that *server* should be empty string.

If registering with PC/U.are.U protection *server* should be string in format "serial_ number@registration_key".

Registered Neurotec.NImages Pro library must be unregistered using Unregister method.

See Also

Unregister | IsRegistered

7.15.1.8.4. Unregister Method

Unregisters Neurotec.NImages Pro library.

```
public static void Unregister();
```

See Also

Register | IsRegistered

7.15.1.9. NMonochromeImage Class

Provides functionality for managing 1-bit monochrome images.

Properties

Item	Gets or sets the color of the specified pixel in NImage object.

Remarks

This class provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to Monochrome.

7.15.1.9.1. NMonochromeImage.Item Property

Gets or sets the color of the specified pixel in NImage object.

```
public bool this[
        uint x,
        uint y
] {get; set;}
```

Parameters

x	The x coordinate of the pixel.
y	The y coordinate of the pixel.

Property value

If pixel is black then gets/sets `false` and if it is white then gets/sets `true`.

See Also

NImage

7.15.1.10. NPixelFormat Struct

Provides functionality for work with pixel format.

Fields

Grayscale	Each pixel value is stored in 8 bits representing 256 shades of gray.
Monochrome	Each pixel value is stored in 1 bit representing either black or white color.
Rgb	Each pixel value is stored in 24 bits consisting of three 8-bit values representing red, green and blue color components.

Remarks

Image pixel format is not limited to these fields. However only these fields are provided for usage with this SDK.

Properties

BitsPerPixel	Gets number of bits used to store a pixel from NPixelFormat Fields.

Methods

CalcRowLongSize	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
CalcRowSize	Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.
Equals	Determines whether the specified Object is equal to the current Object.
GetHashCode	Is intended for a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.
GetRowLongSize	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.
GetRowSize	Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.
IsValid	Checks whether current NPixelFormat value is valid.

7.15.1.10.1. Grayscale Field

Each pixel value is stored in 8 bits representing 256 shades of gray.

```
public static readonly NPixelFormat Grayscale;
```

See Also

NPixelFormat

7.15.1.10.2. Monochrome Field

Each pixel value is stored in 1 bit representing either black or white color.

```
public static readonly NPixelFormat Monochrome;
```

See Also

NPixelFormat

7.15.1.10.3. Rgb Field

Each pixel value is stored in 24 bits consisting of three 8-bit values representing red, green and blue color components.

```
public static readonly NPixelFormat Rgb;
```

See Also

NPixelFormat

7.15.1.10.4. BitsPerPixel Property

Gets number of bits used to store a pixel from NPixelFormat Fields.

```
public uint BitsPerPixel {get;}
```

Property value

A number of bits.

See Also

NPixelFormat Fields

7.15.1.10.5. CalcRowLongSize Methods

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.

7.15.1.10.5.1. CalcRowLongSize (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.

```
public static ulong CalcRowLongSize(
        uint bitCount,
        uint length
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified bits per pixel.

See Also

Copyright © 2006 Neurotechnologija

CalcRowLongSize

7.15.1.10.5.2. CalcRowLongSize (uint, uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel and alignment.

```
public static ulong CalcRowLongSize(
        uint bitCount,
        uint length,
        uint alignment
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified bits per pixel.

See Also

CalcRowLongSize

7.15.1.10.6. CalcRowSize Methods

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.

7.15.1.10.6.1. CalcRowSize (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified bits per pixel.

```
public static uint CalcRowSize(
        uint bitCount,
        uint length
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified bits per pixel.

See Also

CalcRowSize

7.15.1.10.6.2. CalcRowSize (uint, uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified

bits per pixel and alignment.

```
public static uint CalcRowSize(
        uint bitCount,
        uint length,
        uint alignment
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified bits per pixel.

See Also

CalcRowSize

7.15.1.10.7. Equals Method

Determines whether the specified Object is equal to the current Object.

```
public override bool Equals(
        object obj
);
```

Parameters

obj	The Object to compare with the current Object.

Return Values

`true` if the specified Object is equal to the current Object; otherwise, `false`.

See Also

NPixelFormat

7.15.1.10.8. GetHashCode Method

Is intended for a hash function for a particular type. GetHashCode is suitable for use in hashing algorithms and data structures like a hash table.

```
public override int GetHashCode();
```

Return Values

A hash code for the current Object.

7.15.1.10.9. GetRowLongSize Method

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

7.15.1.10.9.1. GetRowLongSize (uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

```
public ulong GetRowLongSize(
        uint length
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified NPixel-Format.

See Also

GetRowLongSize

7.15.1.10.9.2. GetRowLongSize (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat and alignment.

```
public ulong GetRowLongSize(
        uint length,
        uint alignment
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified NPixel-Format.

See Also

GetRowLongSize

7.15.1.10.10. GetRowSize Method

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

7.15.1.10.10.1. GetRowSize (uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat.

```
public uint GetRowSize(
        uint length
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified NPixel-Format.

See Also

GetRowSize

7.15.1.10.10.2. GetRowSize (uint, uint)

Calculates number of bytes needed to store line of specified length of pixels with specified NPixelFormat and alignment.

```
public uint GetRowSize(
        uint length,
        uint alignment
);
```

Return Values

The number of bytes needed to store line of specified length of pixels with specified NPixel-Format.

See Also

GetRowSize

7.15.1.10.11. IsValid Method

Checks whether current NPixelFormat value is valid.

```
public static bool IsValid(
        NPixelFormat value
);
```

Parameters

value	The NPixelFormat object.

Copyright © 2006 Neurotechnologija

Return Values

Returns `true` if the object is valid NPixelFormat, `false` if not.

7.15.1.11. NRgb Struct

Represents an RGB color.

Constructors

NRgb	Initializes a new instance of the NRgb structure.

Properties

Blue	Blue component value of this NRgb.
Green	Green component value of this NRgb.
Red	Red component value of this NRgb.

7.15.1.11.1. NTgb constructor

Initializes a new instance of the NRgb structure.

```
public NRgb(
        byte red,
        byte green,
        byte blue
);
```

Parameters

red	The blue component value of this NRGB.
green	The green component value of this NRGB.
blue	The red component value of this NRGB.

7.15.1.11.2. Blue Property

Blue component value of this NRgb.

```
public byte Blue {get; set;}
```

Property value

The blue component value of this NRGB.

7.15.1.11.3. Green Property

Green component value of this NRgb.

```
public byte Green {get; set;}
```

Property value

The green component value of this NRGB.

7.15.1.11.4. Red Property

Red component value of this NRgb.

```
public byte Red {get; set;}
```

Property value

The red component value of this NRGB.

7.15.1.12. NRgbImage Class

Provides functionality for managing 24-bit RGB images.

Properties

Item	Gets the pixel by index.

Remarks

This class provides advanced functionality, such as individual pixel value retrieval for image with pixel format equal to Rgb.

7.15.1.12.1. NRgbImage.Item Property

Gets the pixel by index.

```
public NRgb this[
        uint x,
        uint y
] {get; set;}
```

Property value

A NRgb structure.

Parameters

x	The x coordinate to get or set.
y	The y coordinate to get or set.

See Also

NRgb

7.15.1.13. Tiff Class

Provides functionality for loading and saving images in TIFF format.

Constructors

Tiff	Initializes a new instance of the `Tiff` class.

Methods

LoadImage	Creates NImage object.

7.15.1.13.1. Tiff Constructor

Initializes a new instance of the `Tiff`.

```
public Tiff();
```

See Also

`Tiff` Class

7.15.1.13.2. LoadImage Method

Creates NImage object.

7.15.1.13.2.1. LoadImage (string)

Creates NImage object from TIFF file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | LoadImage

7.15.1.13.2.2. LoadImage (IntPtr, int)

Creates NImage object from memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage | LoadImage

7.15.1.13.2.3. LoadImage (byte[])

Creates NImage object from byte array.

```
public static NImage LoadImage(
        byte[] buffer
);
```

Copyright © 2006 Neurotechnologija

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | LoadImage

7.15.1.14. Wsq Class

Provides functionality for loading and saving images in WSQ format.

Methods

LoadImage	Creates NImage object.
SaveImage	Saves NImage object.

Constants

DefaultBitRate	Specifies default bit rate (compression level).

7.15.1.14.1. LoadImage Method

Creates NImage object.

7.15.1.14.1.1. LoadImage (string)

Creates NImage object from WSQ file.

```
public static NImage LoadImage(
        string fileName
);
```

Parameters

fileName	A string that contains the name of the file.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

7.15.1.14.1.2. LoadImage (IntPtr, int)

Creates NImage object from memory buffer.

```
public static NImage LoadImage(
        IntPtr buffer,
        int bufferLength
);
```

Parameters

buffer	Pointer to memory buffer.
bufferLength	Size of memory buffer.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

7.15.1.14.1.3. LoadImage (byte[])

Creates NImage object from byte array.

```
public static NImage LoadImage(
        byte[] buffer
);
```

Parameters

buffer	A byte array.

Return Values

A NImage object.

See Also

NImage | LoadImage | SaveImage

7.15.1.14.2. SaveImage Method

Saves NImage object.

7.15.1.14.2.1. SaveImage (NImage image, string fileName)

Saves NImage object to file in WSQ format.

```
public static void SaveImage(
        NImage image,
        string fileName
);
```

Parameters

image	A NImage object.
fileName	A string that contains the name of the file.

See Also

NImage | LoadImage | SaveImage

7.15.1.14.2.2. SaveImage (NImage image, float bitRate, string fileName)

Saves NImage object to file in WSQ format with specified bit rate.

```
public static void SaveImage(
        NImage image,
        float bitRate,
        string fileName
);
```

Parameters

image	A NImage object.
bitRate	A bit rate (compression level).
fileName	A string that contains the name of the file.

Remarks

bitRate can be set to DefaultBitRate (0.75) that corresponds to 15:1 compression. The lower the bit rate is, the higher is the compression level and vice versa. Another common bit rate is 2.25 for 5:1 compression.

See Also

NImage | LoadImage | SaveImage

7.15.1.14.2.3. SaveImage (NImage image)

Saves NImage object to byte array in WSQ format.

```
public static byte[] SaveImage(
        NImage image
);
```

Parameters

image	A NImage object.

Return Values

A byte array.

See Also

NImage | LoadImage | SaveImage

7.15.1.14.2.4. SaveImage (NImage image, float bitRate)

Saves NImage object to byte array in WSQ format with specified bit rate.

```
public static byte[] SaveImage(
        NImage image,
        float bitRate
);
```

Parameters

image	A NImage object.
bitRate	A bit rate (compression level).

Return Values

A byte array.

Copyright © 2006 Neurotechnologija

Remarks

bitRate can be set to DefaultBitRate (0.75) that corresponds to 15:1 compression. The lower the bit rate is, the higher is the compression level and vice versa. Another common bit rate is 2.25 for 5:1 compression.

See Also

NImage | LoadImage | SaveImage

7.15.1.14.2.5. SaveImage (NImage image, Stream stream)

Saves NImage object to stream in WSQ format.

```
public static void SaveImage(
        NImage image,
        Stream stream
);
```

Parameters

image	A NImage object.
stream	The data stream used to save the image.

See Also

NImage | LoadImage | SaveImage

7.15.1.14.2.6. SaveImage (NImage image, float bitRate, Stream stream)

Saves NImage object to stream in WSQ format with specified bit rate.

```
public static void SaveImage(
        NImage image,
        float bitRate,
        Stream stream
);
```

Parameters

image	A NImage object.
bitRate	A bit rate (compression level).
stream	The data stream used to save the image.

Remarks

bitRate can be set to DefaultBitRate (0.75) that corresponds to 15:1 compression. The lower the bit rate is, the higher is the compression level and vice versa. Another common bit rate is 2.25 for 5:1 compression.

See Also

NImage | LoadImage | SaveImage

Copyright © 2006 Neurotechnologija

Appendix A. Support Information

Neurotechnologija provides customer support during the entire period, while the customer develops and uses his own system based on our products. Customers are welcome to contact the technical support (<support@neurotechnologija.com>) for any help on solving the development problems.

Appendix B. Distribution Content

MegaMatcher SDK distribution contains the following folders and files:

bin/Win*/

A directory containing binaries for Microsoft Windows operating system. Currently there is only `Win32_x86` directory for Windows OS running on 32-bit x86 CPU.

`ANTemplate.dll`	ANTemplate library.
`FMRecord.dll`	FMRecord library.
`FPScannerMan.dll`	FPScannerMan library.
`FPScannerManCom.dll`	FPScannerManCom library.
`FPSmm/FPSmmAtmel.dll`	Atmel FingerChip sensor module for FPScannerMan library.
`FPSmm/FPSmmBiometrika.dll`	Biometrika FX2000 scanner module for FPScannerMan library.
`FPSmm/FPSmmCrossMatch.dll`	CrossMatch V300/V300LC scanner module for FPScannerMan library.
`FPSmm/FPSmmFM200.dll`	Startek FM200 scanner module for FPScannerMan library.
`FPSmm/FPSmmIdentix.dll`	Identix DFR2090 scanner module for FPScannerMan library.
`FPSmm/FPSmmLighTunning.dll`	LighTunning LTT-C500 scanner module for FPScannerMan library.
`FPSmm/FPSmmST.dll`	STMicroelectronics TouchChip TCRU1C sensor module for FPScannerMan library.
`FPSmm/FPSmmTacoma.dll`	Tacoma CMOS scanner module for FPScannerMan library.
`FPSmm/FPSmmUareU.dll`	Digital Persona U.are.U 2000/4000 scanner module for FPScannerMan library.
`FPSmm/FingerChip.dll`	Required for `FPSmm/FPSmmAtmel` module.
`FPSmm/FingerPrinterDll.dll`	Required for `FPSmm/FPSmmFM200` module.
`FPSmm/GetImageC500.dll`	Required for `FPSmm/`

	`FPSmmLighTunning` module.
`FPSmm/SMZ_API.dll`	Required for `FPSmm/FPSmmTacoma` module.
`FPSmm/SmzCmos1.dll`	Required for `FPSmm/FPSmmTacoma` module.
`FPSmm/TCI.dll`	Required for `FPSmm/FPSmmST` module.
`FPSmm/fm200api.dll`	Required for `FPSmm/FPSmmFM200` module.
`FPSmm/fm200drv.dll`	Required for `FPSmm/FPSmmFM200` module.
`FPSmm/fx3.dll`	Required for `FPSmm/FPSmmBiometrika` module.
`FPSmm/fx3scan.dll`	Required for `FPSmm/FPSmmBiometrika` module.
`FPSmm/msvcp71.dll`	Microsoft C++ Runtime Library 7.1.
`FPSmm/msvcr71.dll`	Microsoft C Runtime Library 7.1.
`NCore.dll`	NCore library.
`NFExtractor.dll`	NFExtractor library.
`NFMatcher.dll`	NFMatcher library.
`NFRecord.dll`	NFRecord library.
`NFTemplate.dll`	NFTemplate library.
`NImagesPro.dll`	NImages Pro library.
`NMatcher.dll`	NMatcher library.
`NTemplate.dll`	NTemplate library.
`NfsMatcher.dll`	NfsMatcher library.
`VFExtractor.dll`	VFExtractor library.
`VFMatcher.dll`	VFMatcher library.
`admin.exe`	Compiled Admin sample executable.
`logger.exe`	Logger executable.
`msvcp71.dll`	Microsoft C++ Runtime Library 7.1.

Copyright © 2006 Neurotechnologija

msvcr71.dll	Microsoft C Runtime Library 7.1.
mysql_drv.dll	MySQL database connection driver.
oracle_drv.dll	Oracle database connection driver.
node.conf	Sample of configuration file for Node.
node.exe	Node executable.
server.conf	Sample of configuration file for the Server.
server.exe	Server executable.

bin/dotNET/

A directory containing binaries for .NET Framework 1.1 platform.

Neurotec.dll	Neurotec library.
Neurotec.Biometrics.ANTemplate.dll	Neurotec.Biometrics.ANTemplate library.
Neurotec.Biometrics.FMRecord.dll	Neurotec.Biometrics.FMRecord library.
Neurotec.Biometrics.FPScannerMan.dll	Neurotec.Biometrics.FPScannerMan library.
Neurotec.Biometrics.Gui.NFView.dll	Neurotec.Biometrics.Gui.NFView library.
Neurotec.Biometrics.NFExtractor.dll	Neurotec.Biometrics.NFExtractor library.
Neurotec.Biometrics.NFMatcher.dll	Neurotec.Biometrics.NFMatcher library.
Neurotec.Biometrics.NFRecord.dll	Neurotec.Biometrics.NFRecord library.
Neurotec.Biometrics.NFTemplate.dll	Neurotec.Biometrics.NFTemplate library.
Neurotec.Biometrics.NMatcher.dll	Neurotec.Biometrics.NMatcher library.
Neuro-	Neurotec.Biometrics.NTemplate library.

tec.Biometrics.NTemplate.dll	
Neuro-tec.Biometrics.NfsMatcher.dll	Neurotec.Biometrics.NfsMatcher library.
Neuro-tec.Biometrics.VFExtractor.dll	Neurotec.Biometrics.VFExtractor library.
Neuro-tec.Biometrics.VFMatcher.dll	Neurotec.Biometrics.VFMatcher library.
Neurotec.Images.dll	Neurotec.Images Pro library.

bin/linux_*/

A directory containing binaries for Linux-based operating systems. Currently there is only linux_x86 directory for Linux OS running on 32-bit x86 CPU.

admin	Compiled Admin sample executable.
client	Compiled Client sample executable.
fngr2sql	Compiled sample of MySQL database tool executable.
logger	Logger executable.
node	Node executable.
node.conf	Sample of configuration file for Node.
server	Server executable.
server.conf	Sample of configuration file for the Server.

documentation/

A directory containing documentation.

MegaMatcher SDK.chm	This documentation in CHM format.
MegaMatcher SDK.pdf	This documentation in PDF format.

include/

A directory containing header files. It has subdirectories - `Windows` and `linux` which contain the same files but with appropriate line-ends for Windows and Linux operating system accordingly.

`ANTemplate.h`	Header file for ANTemplate module.
`Bmp.h`	Header file for Bmp module.
`FMRecord.h`	Header file for FMRecord module.
`FmrFingerView.h`	Header file for FmrFingerView module.
`IHead.h`	Header file for IHead module.
`NCore.h`	Header file for NCore module.
`NErrors.h`	Header file for NErrors module.
`NFExtractor.h`	Header file for NFExtractor module.
`NFExtractorParams.h`	Required for `NFExtractor.h` header.
`NFExtractorTypes.h`	Required for `NFExtractor.h` header.
`NFMatcher.h`	Header file for NFMatcher module.
`NFMatcherParams.h`	Required for `NFMatcher.h` header.
`NFRecord.h`	Header file for NFRecord module.
`NFRecordV1.h`	Additional header file for NFRecord module.
`NFTemplate.h`	Header file for NFTemplate module.
`NGrayscaleImage.h`	Header file for NGrayscaleImage module.
`NImage.h`	Required for `NImagePro.h`.
`NImageFile.h`	Header file for NImageFile module.
`NImageFormat.h`	Required for `NImageFormatPro.h`.
`NImageFormatPro.h`	Header file for NImageFormat Pro module.
`NImagePro.h`	Header file for NImage Pro module.
`NImages.h`	Required for `NImagesPro.h`.
`NImagesPro.h`	Header file for NImages Pro module.
`NMMatchDetails.h`	Required for `NMatcher.h` header.
`NMatcher.h`	Header file for NMatcher module.

Copyright © 2006 Neurotechnologija

NMatcherParams.h	Required for NMatcher.h header.
NMonochromeImage.h	Header file for NMonochromeImage module.
NParameters.h	Header file for NParameters module.
NPixelFormat.h	Header file for NPixelFormat module.
NRgbImage.h	Header file for NRgbImage module.
NTemplate.h	Header file for NTemplate module.
NTypes.h	Header file for NTypes module.
NfmMatchDetails.h	Required for NFMatcher.h header.
NfsMatcher.h	Header file for NfsMatcher module.
NfsMatcherParams.h	Required for NfsMatcher.h header.
NfsmMatchDetails.h	Required for NfsMatcher.h header.
Tiff.h	Header file for Tiff module.
VFExtractor.h	Header file for VFExtractor module.
VFExtractorParams.h	Required for VFExtractor.h header.
VFExtractorTypes.h	Required for VFExtractor.h header.
VFMatcher.h	Header file for VFMatcher module.
VFMatcherParams.h	Required for VFMatcher.h header.
VfmMatchDetails.h	Required for VFMatcher.h header.
Wsq.h	Header file for Wsq module.

install/

A directory containg files to install.

install/Fingerprint Scanners/	A directory containing drivers for fingerprint scanners.
install/LicenseManager/	A directory containing LAN License Manager installation files for Windows and Linux in accordingly Windows and linux subdirectories.

Copyright © 2006 Neurotechnologija

lib/Win*/

A directory contains library and/or DLL import library files for Microsoft Windows operating system. Currently there is only `Win32_x86` directory for Windows OS running on 32-bit x86 CPU.

`ANTemplate.dll.lib`	Import library for ANTemplate library.
`FMRecord.dll.lib`	Import library for FMRecord library.
`NCore.dll.lib`	Import library for NCore library.
`NFExtractor.dll.lib`	Import library for NFExtractor library.
`NFMatcher.dll.lib`	Import library for NFMatcher library.
`NFRecord.dll.lib`	Import library for NFRecord library.
`NFTemplate.dll.lib`	Import library for NFTemplate library.
`NfsMatcher.dll.lib`	Import library for NfsMatcher library.
`NImagesPro.dll.lib`	Import library for NImages Pro library.
`NMatcher.dll.lib`	Import library for NMatcher library.
`NTemplate.dll.lib`	Import library for NTemplate library.
`VFExtractor.dll.lib`	Import library for VFExtractor library.
`VFMatcher.dll.lib`	Import library for VFMatcher library.

lib/linux_*/

A directory containing library files for Linux-based operating systems. Currently there is only `linux_x86` directory for Linux OS running on 32-bit x86 CPU.

`libANTemplate.so`	ANTemplate library.
`libFMRecord.so`	FMRecord library.
`libNCore.so`	NCore library.
`libNFExtractor.so`	NFExtractor library.
`libNFMatcher.so`	NFMatcher library.
`libNFRecord.so`	NFRecord library.
`libNFTemplate.so`	NFTemplate library.
`libNfsMatcher.so`	NfsMatcher library.

Copyright © 2006 Neurotechnologija

libNImagesPro.so	NImages Pro library.
libNMatcher.so	NMatcher library.
libNTemplate.so	NTemplate library.
libVFExtractor.so	VFExtractor library.
libVFMatcher.so	VFMatcher library.
libcommon.so	Common library.
mysql_drv.so	MySQL database connection driver.
oracle_drv.so	Oracle database connection driver.

pots/

A directory containing translation templates for executables.

admin_s.pot	Admin sample messages.
client_s.pot	Client sample messages.
common.pot	Common messages.
fngr2sql.pot	fngr2sql tool messages.
multi_fngr2sql.pot	multi_fngr2sql tool messages.
node.pot	Node messages.
server.pot	Server messages.

samples/

A directory containing samples.

Windows/admin/	A directory containing source of Admin sample for Windows.
dotNET/ MatchingClusterClientSample/	A directory containing source of MatchingClusterClient sample for .NET.
linux/admin/	A directory containing source of Admin sample for Linux.
linux/client/	A directory containing source of Client

Copyright © 2006 Neurotechnologija

	sample for Linux.
`linux/fngr2sql/`	A directory containing source of fngr2sql sample for Linux.

`sql_drivers/`

This directory contain server and node SQL drivers files.

`node_sql_driver`	This directory contains node SQL drivers files.
`node_sql_driver/sqlite`	This directory contains node SQLite DB driver files.
`node_sql_driver/mysql`	This directory contains node MySQL DB driver files.
`server_sql_driver`	This directory contains server SQL drivers files.
`server_sql_driver/Oracle_drv`	This directory contains server Oracle DB driver files.
`server_sql_driver/MySQL_drv`	This directory contains server MySQL DB driver files.
`include`	This directory contains include files for node and server SQL drivers.

Copyright © 2006 Neurotechnologija

Appendix C. Change Log

This appendix lists MegaMatcher SDK components changes among versions.

Legend

- FIX - bug was fixed.
- CHN - some changes were made.
- UPD - something has been updated.
- ADD - something has been added.
- REM - something has been removed.

Version 1.1.0.0

- UPD: NCore library to version 2.0.1.1.
- UPD: Neurotec library to version 2.0.1.1.
- UPD: NImages Pro library to version 2.0.0.1.
- UPD: Neurotec.Images Pro library to version 2.0.0.0.
- UPD: NFRecord library to version 1.1.0.0.
- UPD: Neurotec.Biometrics.NFRecord library to version 1.1.0.0.
- UPD: NFTemplate library to version 1.1.0.0.
- UPD: Neurotec.Biometrics.NFTemplate library to version 1.1.0.0.
- UPD: NTemplate library to version 1.1.0.0.
- UPD: Neurotec.Biometrics.NTemplate library to version 1.1.0.0.
- UPD: FMRecord library to version 1.1.0.0.
- UPD: Neurotec.Biometrics.FMRecord library to version 1.1.0.0.
- UPD: ANTemplate library to version 1.0.0.2.
- UPD: NFExtractor library to version 1.0.2.0.
- UPD: Neurotec.Biometrics.NFExtractor library to version 1.0.2.0.
- UPD: VFExtractor library to version 4.2.1.0.
- UPD: Neurotec.Biometrics.VFExtractor library to version 4.2.1.0.
- UPD: NMatcher library to version 1.0.2.0.
- ADD: Neurotec.Biometrics.NMatcher library version 1.0.1.0.
- ADD: NfsMatcher library version 1.0.0.0.
- ADD: Neurotec.Biometrics.NfsMatcher library version 1.0.0.0.
- ADD: NFMatcher library version 1.0.2.0.
- ADD: Neurotec.Biometrics.NFMatcher library version 1.0.0.0.
- ADD: VFMatcher library version 4.2.1.0.
- ADD: Neurotec.Biometrics.VFMatcher library version 4.2.0.0.
- UPD: FPScannerMan library to version 1.0.0.2.
- UPD: FPSmmAtmel library to version 1.0.0.1.
- UPD: FPSmmBiometrika library to version 1.0.0.1.
- UPD: FPSmmCrossMatch library to version 1.0.0.2.
- UPD: FPSmmFM200 library to version 1.0.0.1.
- UPD: FPSmmIdentix library to version 1.0.0.1.
- UPD: FPSmmLighTunning library to version 1.0.0.1.

- UPD: FPSmmST library to version 1.0.0.1.
- UPD: FPSmmTacoma library to version 1.0.0.1.
- UPD: FPSmmUareU library to version 1.0.0.1.
- UPD: Neurotec.Biometrics.FPScannerMan library to version 1.0.0.2.
- ADD: FPScannerManCom library version 1.0.0.0.
- UPD: Neurotec.Biometrics.Gui.NFView library to version 1.1.0.0.
- UPD: MatchingClusterClient sample to version 1.1.0.0.
- UPD: Node to version 1.0.0.5.
- REM: Template conversion sample. See template management and conversion section for examples.
- ADD: Visual Basic 6 sample client application added.
- UPD: Major updates in components reference (documentation).
- UPD: Node to version 1.1.0.0.
- UPD: Server to version 1.1.0.0.

Version 1.0.0.6

- ADD: Implemented lightweight cluster database update. For more information see Server changes.
- ADD: Remote cluster logging capability. For more information see Server and Node changes.
- UPD: Admin Sample to version 1.0.0.2.
- UPD: Node to version 1.0.0.4.
- UPD: Server to version 1.0.0.4.
- ADD: Logger version 1.0.0.0.
- UPD: NFExtractor library to version 1.0.1.2.
- UPD: MatchingClusterClient sample to version 1.0.0.4.

Version 1.0.0.5

- UPD: NCore library to version 1.0.0.2.
- UPD: Neurotec library to version 1.0.0.2.
- UPD: NFRecord library to version 1.0.1.1.
- UPD: Neurotec.Biometrics.NFRecord library to version 1.0.1.1.
- UPD: NFTemplate library to version 1.0.0.3.
- UPD: Neurotec.Biometrics.NFTemplate library to version 1.0.0.1.
- UPD: NFExtractor library to version 1.0.1.1.
- UPD: Neurotec.Biometrics.NFExtractor library to version 1.0.0.2.
- UPD: VFExtractor library to version 1.0.0.2.
- UPD: Neurotec.Biometrics.NFExtractor library to version 1.0.0.1.
- UPD: NMatcher library to version 1.0.1.1.

Version 1.0.0.4

- UPD: Major changes in documentation.

Copyright © 2006 Neurotechnologija

Version 1.0.0.3

- CHN: Re-enrollment of fingerprint templates is required. For more information see NFExtractor module changes.
- UPD: NCore library to version 1.0.0.1.
- UPD: Neurotec library to version 1.0.0.1.
- UPD: NImagesPro library to version 1.0.0.2.
- UPD: NFRecord library to version 1.0.1.0.
- UPD: Neurotec.Biometrics.NFRecord library to version 1.0.1.0.
- UPD: NFTemplate library to version 1.0.0.2.
- UPD: NTemplate library to version 1.0.0.2.
- UPD: FMRecord library to version 1.0.1.0.
- UPD: Neurotec.Biometrics.FMRecord library to version 1.0.1.0.
- UPD: NFExtractor library to version 1.0.1.0.
- UPD: Neurotec.Biometrics.NFExtractor library to version 1.0.0.1.
- UPD: VFExtractor library to version 1.0.0.1.
- UPD: NMatcher library to version 1.0.1.0.
- UPD: Server to version 1.0.0.3.
- UPD: Node to version 1.0.0.3.
- UPD: MatchingClusterClient sample to version 1.0.0.3.
- UPD: U.are.U scanner drivers (Integrator Gold 2.5.0).
- FIX: Minor fixes in documentation.

Version 1.0.0.2

- UPD: NFRecord library to version 1.0.0.2.
- UPD: Neurotec.Biometrics.NFRecord library to version 1.0.0.1.
- UPD: Server to version 1.0.0.2.
- UPD: Node to version 1.0.0.2.
- UPD: MatchingClusterClient sample to version 1.0.0.2.
- ADD: Change log to documentation.
- FIX: Minor fixes in documentation.

Version 1.0.0.1

- UPD: NImagesPro library to version 1.0.0.1.
- UPD: NFRecord library to version 1.0.0.1.
- UPD: NFTemplate library to version 1.0.0.1.
- UPD: NTemplate library to version 1.0.0.1.
- UPD: FMRecord library to version 1.0.0.1.
- UPD: NFExtractor library to version 1.0.0.1.
- UPD: NMatcher library to version 1.0.0.1.
- UPD: FPScannerMan library to version 1.0.0.1.
- UPD: FPSmmCrossMatch library to version 1.0.0.1.
- UPD: Neurotec.Biometrics.FPScannerMan library to version 1.0.0.1.
- UPD: MatchingClusterClient sample to version 1.0.0.1.
- UPD: Server to version 1.0.0.1.

Copyright © 2006 Neurotechnologija

- UPD: Node to version 1.0.0.1.
- UPD: Admin sample to version 1.0.0.1.
- FIX: Minor fixes in documentation

Version 1.0.0.0

Initial release. Contains the following components:

- ANTemplate library version 1.0.0.0.
- Admin sample version 1.0.0.0.
- FMRecord library version 1.0.0.0.
- FPScannerMan library version 1.0.0.0.
- FPSmmAtmel library version 1.0.0.0.
- FPSmmBiometrika library version 1.0.0.0.
- FPSmmCrossMatch library version 1.0.0.0.
- FPSmmFM200 library version 1.0.0.0.
- FPSmmIdentix library version 1.0.0.0.
- FPSmmLighTunning library version 1.0.0.0.
- FPSmmST library version 1.0.0.0.
- FPSmmTacoma library version 1.0.0.0.
- FPSmmUareU library version 1.0.0.0.
- MatchingClusterClient sample version 1.0.0.0.
- MySQL driver version 1.0.0.0.
- NCore library version 1.0.0.0.
- NFExtractor library version 1.0.0.0.
- NFRecord library version 1.0.0.0.
- NFTemplate library version 1.0.0.0.
- NImages Pro library version 1.0.0.0.
- NMatcher library version 1.0.0.0.
- NTemplate library version 1.0.0.0.
- Neurotec library version 1.0.0.0.
- Neurotec.Biometrics.ANTemplate library version 1.0.0.0.
- Neurotec.Biometrics.FMRecord library version 1.0.0.0.
- Neurotec.Biometrics.FPScannerMan library version 1.0.0.0.
- Neurotec.Biometrics.Gui.NFView library version 1.0.0.0.
- Neurotec.Biometrics.NFExtractor library version 1.0.0.0.
- Neurotec.Biometrics.NFRecord library version 1.0.0.0.
- Neurotec.Biometrics.NFTemplate library version 1.0.0.0.
- Neurotec.Biometrics.NTemplate library version 1.0.0.0.
- Neurotec.Biometrics.VFExtractor library version 1.0.0.0.
- Neurotec.Images Pro library version 1.0.0.0.
- Node version 1.0.0.0.
- Server to version 1.0.0.0.
- Template conversion sample.
- VFExtractor library version 1.0.0.0.

C.1. ANTemplate Library

Copyright © 2006 Neurotechnologija

Version 1.0.0.2

- UPD: Minor updates.

Version 1.0.0.1

- UPD: Minor updates.

Version 1.0.0.0

Initial release.

C.2. Admin Sample

Version 1.0.0.2

- ADD: Added cluster information retrieve capability.
- ADD: Added lightweight database update capability.

Version 1.0.0.1

- FIX: Dependence on Visual C++ runtime library on Win32_x86 platform.

Version 1.0.0.0

Initial release.

C.3. FMRecord Library

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.2.0

- CHN: Some changes in interface.
- ADD: Support for Neurotechnologija data.

Version 1.0.1.0

- ADD: Four-neighbours ridge count support.
- CHN: Flags definition.

Copyright © 2006 Neurotechnologija

Version 1.0.0.1

- CHN: Flags definition.

Version 1.0.0.0

Initial release.

C.4. FPScannerMan Library

Version 1.0.0.2

- UPD: Minor updates.

Version 1.0.0.1

- FIX: Internal synchronization logic.

Version 1.0.0.0

Initial release.

C.5. FPScannerManCom Library

Version 1.0.0.0

Initial release.

C.6. FPSmmAtmel Library

Version 1.0.0.1

- UPD: Minor updates

Version 1.0.0.0

Initial release.

C.7. FPSmmBiometrika Library

Version 1.0.0.1

- UPD: Minor updates

Version 1.0.0.0

Initial release.

C.8. FPSmmCrossMatch Library

Version 1.0.0.2

- UPD: Minor updates

Version 1.0.0.1

- FIX: Errors with synchronization logic

Version 1.0.0.0

Initial release.

C.9. FPSmmFM200 Library

Version 1.0.0.1

- UPD: Minor updates

Version 1.0.0.0

Initial release.

C.10. FPSmmIdentix Library

Version 1.0.0.1

- UPD: Minor updates

Version 1.0.0.0

Initial release.

C.11. FPSmmLighTunning Library

Version 1.0.0.1

- UPD: Minor updates

Version 1.0.0.0

Initial release.

Copyright © 2006 Neurotechnologija

C.12. FPSmmST Library

Version 1.0.0.1

- UPD: Minor updates

Version 1.0.0.0

Initial release.

C.13. FPSmmTacoma Library

Version 1.0.0.1

- UPD: Minor updates

Version 1.0.0.0

Initial release.

C.14. FPSmmUareU Library

Version 1.0.0.1

- UPD: Minor updates

Version 1.0.0.0

Initial release.

C.15. Logger

Version 1.0.0.0

Initial release.

C.16. MatchingClusterClient Sample

Version 1.1.0.0

- ADD: Latent fingerprint record editor.
- ADD: Multi-finger template generation capability.
- UPD: Update with new versions of components.

Version 1.0.0.4

- ADD: Sends a message to server when new entry enrolled. Then nodes are informed of database changes.

Version 1.0.0.3

- ADD: Extracted ridge counts parameter to extractor.
- ADD: Medium speed value for speed parameter for NF fingerprint matcher.
- CHN: Settings are saved in version independent manner.

Version 1.0.0.2

- FIX: Minor fixes.

Version 1.0.0.1

- FIX: Sockets not shutting down in some cases.
- ADD: Matching speed and maximal rotation parameters for NF fingerprint matcher.

Version 1.0.0.0

Initial release.

C.17. MySQL Driver

Version 1.0.0.0

Initial release.

C.18. NCore Library

Version 2.0.1.1

- CHN: Minor changes.

Version 2.0.1.0

- ADD: NParameters module instead of NMetaTypes module for internal infrastructure support.
- CHN: Infrastructure optimization for 64-bit support.

Version 2.0.0.0

- ADD: A lot of stuff for internal infrastructure support.
- CHN: Some changes in internal infrastructure support.

Version 1.0.0.2

- ADD: NIndexPair structure.

Version 1.0.0.1

- FIX: Minor fixes in headers.

Version 1.0.0.0

Initial release.

C.19. NFExtractor Library

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 neighbours extraction logic. Re-enrollment is required for NMatcher to work properly.
- ADD: NFEP_EXTRACTED_RIDGE_COUNTS parameter (can now store indexes of minutiae neighbours in template and it is now the default behavior).
- UPD: U.are.U scanner mode.

Version 1.0.0.1

- FIX: NfeExtract function division by zero with some images.

Version 1.0.0.0

Initial release.

C.20. NFMatcher Library

Version 1.0.2.0

Initial release, separated from NMatcher library.

C.21. NFRecord Library

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.2.0

- ADD: Packing to version 1.0 functionality.

Version 1.0.1.1

- UPD: Minor updates.

Version 1.0.1.0

- UPD: Made compatible with new functionality of NFExtractor library version 1.0.1.0.
- ADD: Four-neighbours ridge count support.
- CHN: Flags definition.
- FIX: Minor fixes.

Version 1.0.0.2

- FIX: NFSelectMinutiaeNeighbours parameter list.

Version 1.0.0.1

- FIX: NFRecordGetPositionMem, NFRecordGetImpressionTypeMem, NFRecordGetPatternClassMem, NFRecordGetQualityMem, NFRecordGetGMem functions retrieve correct value from packed NFRecord.
- CHN: NFSelectMinutiaeNeighbours takes flags as last parameter.
- ADD: Additional flag.

Version 1.0.0.0

Initial release.

C.22. NFTemplate Library

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.0.4

- UPD: Minor updates.

Version 1.0.0.3

- ADD: NFTemplateGetRecordCountMem function.
- FIX: Minor fixes.

Version 1.0.0.2

- FIX: Minor fixes.

Version 1.0.0.1

- FIX: NFTemplateUnpack function returned error in some cases.

Version 1.0.0.0

Initial release.

C.23. NImages Pro Library

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.

C.24. NMatcher Library

Version 1.0.2.0

- CHN: NfsMatcher, NFMatcher and VFMatcher separated from library.
- CHN: Used fingerprint matcher parameter moved to NfsMatcher.
- CHN: Some changes in interface and match details.

Version 1.0.1.1

- ADD: NMVerifyEx and NMIdentifyEx functions added that takes pointer to match details (mated minutiae for fingerprints).
- FIX: Minor fixes.

Version 1.0.1.0

- UPD: Fingerprint matching is made compatible with new functionality of NFExtractor library version 1.0.1.0.
- ADD: NFM_SPEED_MEDIUM value for NFMP_MATCHING_SPEED parameter.
- CHN: NFM_SPEED_MEDIUM corresponds to NFM_SPEED_HIGH in previous version.
- FIX: Fingerprint matching speed optimizations.

Version 1.0.0.1

- FIX: Multiple fingerprint matching logic.
- ADD: NFMP_MATCHING_SPEED and NFMP_MAXIMAL_ROTATION parameters.
- FIX: Fingerprint matching speed optimizations.

Version 1.0.0.0

Initial release.

C.25. NTemplate Library

Version 1.1.0.0

- CHN: Some changes in interface.
- ADD: Editing support.

Version 1.0.0.3

- UPD: Minor updates.

Version 1.0.0.2

- FIX: Minor fixes.

Version 1.0.0.1

- FIX: Signature of packed NTemplate.
- FIX: NTemplateUnpack function returned error in some cases.
- FIX: NTemplateUnpack logic.

Version 1.0.0.0

Initial release.

C.26. Neurotec Library

Version 2.0.1.1

- FIX: Minor fixes.
- UPD: Minor updates in structures.

Version 2.0.1.0

- ADD: NParameters class for internal infrastructure support.
- CHN: Infrastructure optimization for 64-bit support.

Version 2.0.0.0

- ADD: A lot of stuff for internal infrastructure support.
- CHN: Some changes in internal infrastructure support.

Version 1.0.0.2

- ADD: NIndexPair structure.

Version 1.0.0.1

- FIX: All error codes are mapped to appropriate exceptions.

Version 1.0.0.0

Initial release.

C.27. Neurotec.Biometrics.ANTemplate Library

Version 1.0.0.0

Initial release.

C.28. Neurotec.Biometrics.FMRecord Library

Version 1.1.0.0

- ADD: Editing support.

Version 1.0.2.0

- ADD: Neurotechnologija data support.

Version 1.0.1.0

- ADD: Four-neighbours ridge count support.

Version 1.0.0.0

Initial release.

C.29. Neurotec.Biometrics.FPScannerMan Library

Version 1.0.0.2

- UPD: Minor updates.

Version 1.0.0.1

- FIX: Shutdown errors.

Version 1.0.0.0
Initial release.

C.30. Neurotec.Biometrics.Gui.NFView Library

Version 1.1.0.0

- ADD: Events enabling minutiae editing.
- ADD: Minutia selection and neighbours displaying capability.
- CHN: Some changes in internal logic.

Version 1.0.0.1

- FIX: Minor fixes.

Version 1.0.0.0
Initial release.

C.31. Neurotec.Biometrics.NFExtractor Library

Version 1.0.2.0

- ADD: Extraction of unpacked template capability.
- ADD: Update of edited template capability.
- CHN: Parameters interface.

Version 1.0.0.2

- FIX: Minor fixes.

Version 1.0.0.1

- ADD: ParameterExtractedRidgeCounts and ExtractedRidgeCounts property that wraps it. For more information see NFExtractor library changes in version 1.0.1.0.

Version 1.0.0.0
Initial release.

C.32. Neurotec.Biometrics.NFMatcher Library

Version 1.0.0.0
Initial release, separated from Neurotec.Biometrics.NMatcher library.

C.33. Neurotec.Biometrics.NFRecord Library

Version 1.1.0.0

- ADD: Editing support.

Version 1.0.2.0

- ADD: Packing to version 1.0 functionality.

Version 1.0.1.1

- UPD: Minor updates.

Version 1.0.1.0

- ADD: Four-neighbours ridge count support.

Version 1.0.0.1

- FIX: NFSelectMinutiaeNeighbours parameter list.

Version 1.0.0.0
Initial release.

C.34. Neurotec.Biometrics.NFTemplate Library

Version 1.1.0.0

- ADD: Editing support.

Version 1.0.0.2

- UPD: Minor updates.

Version 1.0.0.1

- ADD: GetRecordCount static method.

Version 1.0.0.0
Initial release.

C.35. Neurotec.Biometrics.NMatcher Library

Version 1.0.1.0

- CHN: NfsMatcher, NFMatcher and VFMatcher separated from library.
- CHN: Used fingerprint matcher parameter moved to NfsMatcher.
- CHN: Some changes in interface and match details.

Version 1.0.0.0
Initial release.

C.36. Neurotec.Biometrics.NTemplate Library

Version 1.1.0.0

- ADD: Editing support.

Version 1.0.0.1

- UPD: Minor updates.

Version 1.0.0.0
Initial release.

C.37. Neurotec.Biometrics.NfsMatcher Library

Version 1.0.0.0
Initial release, separated from Neurotec.Biometrics.NMatcher library.

C.38. Neurotec.Biometrics.VFExtractor Library

Version 4.2.1.0

- ADD: Extraction of unpacked template capability.
- CHN: Parameters interface.
- CHN: Major version number made consistent with VeriFinger major version.

Version 1.0.0.1

- FIX: Minor fixes.

Version 1.0.0.0

Initial release.

C.39. Neurotec.Biometrics.VFMatcher Library

Version 4.2.0.0

Initial release, separated from Neurotec.Biometrics.NMatcher library.

C.40. Neurotec.Images Pro Library

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.

C.41. NfsMatcher Library

Version 1.0.0.0

Initial release, separated from NMatcher library.

C.42. Node

Version 1.1.0.0

- ADD: Node uses SQLite database.

Version 1.0.0.5

- UPD: Minor updates.

Version 1.0.0.4

- ADD: Added remote node logging capability.

Version 1.0.0.3

- FIX: Minor fixes.

Version 1.0.0.2

- FIX: License not released in some cases.
- FIX: Power calculation.

Version 1.0.0.1

- FIX: Dependence on Visual C++ runtime library on Win32_x86 platform.

Version 1.0.0.0

Initial release.

C.43. Server

Version 1.1.0.0

- ADD: Oracle database was joined to the server. It is posible to use other providers database.

Version 1.0.0.4

- ADD: Implemented lightweight database update.
- ADD: Added cluster information retrieval capability.
- ADD: Added remote logging capability.

Version 1.0.0.3

- FIX: Minor fixes.

Version 1.0.0.2

- FIX: License not released in some cases.

Version 1.0.0.1

- FIX: Dependence on Visual C++ runtime library on Win32_x86 platform.

Version 1.0.0.0

Initial release.

C.44. VFExtractor Library

Version 4.2.1.0

- ADD: Extraction of unpacked template capability.
- UPD: Minor updates.
- CHN: Major version number made consistent with VeriFinger major version.

Version 1.0.0.2

- FIX: Minor fixes.

Version 1.0.0.1

- UPD: U.are.U scanner mode.

Version 1.0.0.0

Initial release.

C.45. VFMatcher Library

Version 4.2.1.0

Initial release, separated from NMatcher library.

Glossary

Cluster	A group of computers connected by network and dedicated to one task.
DBMS	Database Management System.
Server side components	Server software, node software, Matcher module, database and administrating tools.
SDK	Software Development Kit.
Task	The data(template and query) provided by the user is wrapped into a single unit that is called a task.
WSQ	Wavelet Scalar Quantization algorithm. Compresses fingerprint images up to 10-15 times.
LSBS	Large-scale Biometric System
Main database	Database, where all of persons' biometric templates are stored.
AN2K	ANSI/NIST ITL-1-2000 Standard. Data format interchange of Fingerprint, Facial, & Scar Mark & Tattoo (SMT) Information
ANSI378	ANSI/INCIST 378 2004 Standard. Finger Minutia Format for Data Interchange.
OS	Operating system.