UCBioBSP SDK

Union Community's Software Development Kit for Biometric Application

Programmer's Guide

Version 3.xx Rev 01

UNION COMMUNITY Co., Ltd.

Copyright © 2008, UNION COMMUNITY Co., Ltd.
All rights reserved.

USER License Agreement for Software Developer's Kit Designed by Union Community Co., Ltd

This agreement is a legal usage license agreement between Union Community Co., Ltd. and the user. If you do not agree with the terms and condition of the agreement, please return the product promptly. If you return the product, you will receive a refund.

1. Usage License
UNION COMMUNITY Co., Ltd. Grants licensee to use this SDK a personal, Limited, non-transferable, non-exclusive right to install and use one copy of the SDK on a single computer exclusively.
The software is considered 'being used' if it is stored in a computer's main or other storage device. The number of software copies will be determined by taking the greater number of the number of computers 'used' by the software and the number of computers with the software stored.
Licensee may use the SDK solely for developing, designing, and testing UNION software applications for use with UNION products ("Applications").

2. Right to Upgrade
If you have purchased the software by upgrading an older version, the usage license of the old version is transferred to the new version. However, you may only use the old version under the condition that the old and new versions are not running simultaneously. Therefore, you are prohibited from transferring, renting or selling the old version. You maintain the usage license for the program and ancillary files that are in the old version but not in the new version.

3. Assignment of License
If you wish to transfer the usage license of this software to a third party, you must first obtain a written statement indicating that the recipient agrees with this agreement. You must then transfer the original disk and all other program components, and all copies of the program must be destroyed. After the transfer is complete, you must notify UNION COMMUNITY Co., Ltd. to update the customer registration.
Licensee shall not rent, lease, sell or lend the software application developed using the SDK to a third party without UNION's prior written consent.
Licensee shall not copy and redistribute the SDK without UNION's prior written consent.
No other uses and/or distribution of the SDK or Sample Code are permitted without UNION's prior written consent. UNION reserves all rights not expressly granted to Licensee.

4. Copyright
All copyrights and intellectual properties of the software and its components belong to UNION COMMUNITY Co., Ltd. and these rights are protected under Korean and international copyright laws. Therefore, you may not make copies of the software other than for your backup purposes. In addition,

you may not modify the software other than for reverse-engineering purposes to secure compatibility. Finally, you may not modify, transform or copy any part of the documentation without written permission from UNION COMMUNITY. (If you're using a network product, you may copy the documentation in the amount of the number of users)

5. Installation

An individual user can install this software in his/her PCs at home and office, as well as in a mobile PC. However, the software must not be running from two computers simultaneously. A single product can be installed in two or more computers in one location, but one of those computers must have a usage rate of at least 70%. If another computer has a usage rate of 31% or higher, another copy of the software must be purchased.

6. Limitation of Warranty

UNION COMMUNITY Co., Ltd. guarantees that the CD-ROM and all components are free of physical damage for a year after purchase.
UNION DISCLAIMS ALL WARRANTIES NOT EXPRESSLY PROVIDED IN THIS AGREEMENT INCLUDING, WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE. If you find any manufacture defect within the warranty period, we will replace the product. You must be able to prove that the product has been purchased within a year to receive a replacement, but we will not replace a product damaged due to your mishandling or negligence. UNION COMMUNITY Co., Ltd. does not guarantee that the software and its features will satisfy your specific needs, and is not liable for any consequential damages arising out of the use of this product.

7. Liabilities

UNION COMMUNITY Co., Ltd. is not liable for any verbal, written or other agreements made by third parties, including product suppliers and dealers.

8. Termination

This agreement is valid until the date of termination. However, the agreement shall terminate automatically if you damage the program or its components, or fail to comply with the terms described in this agreement.

9. Customer Service

UNION COMMUNITY Co., Ltd. makes every effort to provide registered customers with technical assistance and solutions to problems regarding software applications under certain system environments. When a customer submits a suggestion about any inconvenience or anomaly experienced during product usage, UNION COMMUNITY Co., Ltd. will take corrective action and notify the customer of the result.

10. General Terms

You acknowledge that you have read, understood and agree with the terms of this agreement. You also recognize the fact that this agreement has precedence over user agreements of older versions, past order agreements, advertisement notifications and/or other written agreements.

11. Contact

If you have any questions about this agreement, please contact UNION COMMUNITY Co., Ltd. via telephone, fax or e-mail.

목 차

1. Overview 20

1.1. SDK의 특징 21
1.2. 제공 모듈 22
1.3. 개발 모델 23
1.4. 지문 데이터 구조 24
1.4.1. Format 24
1.4.2. Header 24
1.4.3. Data Block 26
1.5. 용어 설명 27

2. Installation 28

2.1. 시스템 요구 사항 28
2.2. 설치하기 29
2.3. 설치되는 파일 및 폴더 설명 36
2.3.1. Windows System32 폴더 36
2.3.2. GAC(Global Assembly Cache) 폴더 36
2.3.3. (설치폴더)\Inc 36
2.3.4. (설치폴더)\Lib 37
2.3.5. (설치폴더)\Bin 37
2.3.6. (설치폴더)\dotNET 37
2.3.7. (설치폴더)\dotNet\Setup 38
2.3.8. (설치폴더)\Samples 38
2.3.9. (설치폴더)\Skins 38

3. Programming by using DLL 39

3.1. 함수 호출 구조 39
3.2. 초기화 및 종료 40
3.2.1. 초기화 하기 40
3.2.2. 종료 하기 40

3.3. 기본 설정 41
3.3.1. SDK 버전 얻기 41
3.3.2. 기본 설정 값 얻어오기 및 새로운 값 설정 41
3.3.3. 입력 지문에 대한 품질 설정 43
3.4. 디바이스 사용하기 44
3.4.1. 디바이스 목록 얻기 44
3.4.2. 디바이스 열기 45
3.4.3. 디바이스 닫기 45
3.4.4. 디바이스 정보 얻기 46
3.5. FIR 데이터의 이해 47
3.5.1. FIR의 종류 47
3.5.2. FIR의 사용 47
3.5.3. FIR 메모리 해제 49
3.5.4. FIR의 변환 49
3.6. 지문 등록하기 51
3.6.1. 신규 지문 등록 및 기존 지문 수정 51
3.6.2. Payload 지정 51
3.6.3. 자동 지문 감지(Auto Detect) 51
3.6.4. 사용 예 51
3.7. 지문 획득하기 53
3.7.1. 지문 획득 53
3.7.2. 사용 예 53
3.8. 지문 인증하기 54
3.8.1. 라이브 지문과 등록 지문과의 인증 54
3.8.2. 미리 획득한 지문과 등록 지문과의 인증 54
3.8.3. Payload 데이터 얻기 55
3.9. FastSearch (1:N 인증) 사용하기 57
3.9.1. 초기화 및 종료 57
3.9.2. 기본 설정 값 얻어오기 및 새로운 값 설정 57
3.9.3. DB 만들기 57
3.9.4. 메모리 DB 관리 58
3.9.5. 1:N 인증하기 59
3.10. FIR 데이터 변환하기 61
3.10.1. FIR 데이터로부터 Template 데이터 추출 61
3.10.2. Template 데이터를 이용해 FIR Handle 만들기 61

3.10.3. Template 데이터 간의 변환 63
3.10.4. Audit FIR 데이터로부터 Raw 이미지 추출 63
3.10.5. Raw 이미지를 이용해 Audit FIR Handle 만들기 64
3.11. UI 설정하기 65
3.11.1. Skin 파일 불러오기 65
3.11.2. UI 속성 변경하기 65
3.11.3. Callback 사용하기 67
3.11.4. 사용 예 69
3.12. Smart Card 사용하기 70
3.12.1. 스마트카드의 개요 70
3.12.2. 스마트카드의 RF Power 켜기와 끄기 73
3.12.3. 스마트카드의 Serial 값 읽기 74
3.12.4. Block 값 읽기 및 쓰기 74
4. Programming by using COM 76
4.1. COM 사용 개요 76
4.1.1. COM의 등록 76
4.2. 초기화 및 종료 77
4.2.1. 초기화 하기 77
4.2.2. 종료 하기 77
4.2.3. 하위 인터페이스 선언 77
4.3. 기본 설정 80
4.3.1. SDK 버전 얻기 80
4.3.2. 기본 설정 값 얻어오기 및 새로운 값 설정 80
4.3.3. 입력 지문에 대한 품질 설정 81
4.4. 디바이스 사용하기 83
4.4.1. 디바이스 목록 얻기 83
4.4.2. 디바이스 열기 84
4.4.3. 디바이스 닫기 85
4.4.4. 디바이스 정보 얻기 85
4.5. FIR 데이터의 이해 87
4.5.1. FIR의 종류 87
4.5.2. FIR의 사용 87
4.5.3. FIR 메모리 해제 87

4.6. 지문 등록하기 88
4.6.1. 신규 지문 등록 및 기존 지문 수정 88
4.6.2. Payload 지정 88
4.6.3. 자동 지문 감지(Auto Detect) 88
4.6.4. 사용 예 88
4.7. 지문 획득하기 90
4.7.1. 지문 획득 90
4.7.2. 사용 예 90
4.8. 지문 인증하기 92
4.8.1. 라이브 지문과 등록 지문과의 인증 92
4.8.2. 미리 획득한 지문과 등록 지문과의 인증 92
4.8.3. Payload 데이터 얻기 94
4.9. FastSearch (1:N 인증) 사용하기 95
4.9.1. 초기화 및 종료 95
4.9.2. 기본 설정 값 얻어오기 및 새로운 값 설정 95
4.9.3. DB 만들기 96
4.9.4. 메모리 DB 관리 97
4.9.5. 1:N 인증하기 97
4.10. FIR 데이터 변환하기 100
4.10.1. FIR 데이터로부터 Template 데이터 추출 100
4.10.2. Template 데이터를 이용해 FIR 만들기 100
4.10.3. Audit FIR 데이터로부터 Raw 이미지 추출 101
4.11. UI 설정하기 103
4.11.1. Skin 파일 불러오기 103
4.11.2. UI 속성 변경하기 103
4.11.3. Callback Event Handler사용하기 104
4.12. Smart Card 사용하기 106
4.12.1. 초기화와 종료 106
4.12.2. 스마트카드의 RF Power 켜기와 끄기 106
4.12.3. 스마트카드의 Serial 값 읽기 107
4.12.4. Block 값 읽기 및 쓰기 108

5. API Reference for DLL 110

5.1. Type definitions 110

5.1.1. Basic types 110
- UCBioAPI_SINT8 / UCBioAPI_SINT16 / UCBioAPI_SINT32 110
- UCBioAPI_UINT8 / UCBioAPI_UINT16 / UCBioAPI_UINT32 / UCBioAPI_UINT64 110
- UCBioAPI_SINT / UCBioAPI_UINT 110
- UCBioAPI_VOID_PTR 110
- UCBioAPI_BOOL 110
- UCBioAPI_CHAR / UCBioAPI_CHAR_PTR 110
- UCBioAPI_NULL 110
- UCBioAPI_HWND 110

5.1.2. General types 111
- UCBioAPI_FIR_VERSION 111
- UCBioAPI_VERSION 111
- UCBioAPI_FIR_DATA_TYPE 111
- UCBioAPI_FIR_PURPOSE 112
- UCBioAPI_FIR_QUALITY 112
- UCBioAPI_FIR_PRIVILEGE 112
- UCBioAPI_FIR_DATE 113
- UCBioAPI_FIR_UUID_INFO 113
- UCBioAPI_OPTIONAL_DATA_TYPE 113
- UCBioAPI_FIR_OPTIONAL_DATA 114
- UCBioAPI_FIR_HEADER 115
- UCBioAPI_FIR_DATA 115
- UCBioAPI_FIR_FORMAT 116
- UCBioAPI_FIR 116
- UCBioAPI_FIR_PAYLOAD 116
- UCBioAPI_HANDLE / UCBioAPI_HANDLE_PTR 117
- UCBioAPI_FIR_HANDLE / UCBioAPI_FIR_HANDLE_PTR 117
- UCBioAPI_FIR_SECURITY_LEVEL 117
- UCBioAPI_TEMPLATE_FORMAT 118
- UCBioAPI_INIT_INFO_0 118
- UCBioAPI_DEVICE_ID 119
- UCBioAPI_DEVICE_NAME 120
- UCBioAPI_DEVICE_INFO_0 120
- UCBioAPI_DEVICE_INFO_EX 121
- UCBioAPI_RETURN 122
- UCBioAPI_FIR_TEXTENCODE 122
- UCBioAPI_INPUT_FIR_FORM 122
- UCBioAPI_INPUT_FIR 123
- UCBioAPI_WINDOW_CALLBACK_PARAM_0 123
- UCBioAPI_WINDOW_CALLBACK_PARAM_1 124
- UCBioAPI_WINDOW_CALLBACK_0 / UCBioAPI_WINDOW_CALLBACK_1 125

- UCBioAPI_CALLBACK_INFO_0 / UCBioAPI_CALLBACK_INFO_1 125
- UCBioAPI_WINDOW_STYLE 126
- UCBioAPI_WINDOW_OPTION 127
- UCBioAPI_WINDOW_OPTION_2 128
- UCBioAPI_TEMPLATE_TYPE 129
- UCBioAPI_FINGER_ID 129
- UCBioAPI_MATCH_OPTION_0 130

5.1.3. Export/Import functions related types 131

- UCBioAPI_TEMPLATE_BLOCK 131
- UCBioAPI_FINGER_BLOCK 131
- UCBioAPI_EXPORT_DATA 132
- UCBioAPI_IMAGE_DATA 132
- UCBioAPI_AUDIT_DATA 133
- UCBioAPI_EXPORT_AUDIT_DATA 133

5.1.4. FastSearch functions related types 135

- UCBioAPI_FASTSEARCH_INIT_INFO_0 135
- UCBioAPI_FASTSEARCH_INIT_INFO_0 136
- UCBioAPI_FASTSEARCH_SAMPLE_INFO 136
- UCBioAPI_FASTSEARCH_CALLBACK_PARAM_0 137
- UCBioAPI_FASTSEARCH_CALLBACK_0 137
- UCBioAPI_FASTSEARCH_CALLBACK_INFO_0 138

5.1.5. SmartCard functions related types 139

- UCBioAPI_SC_USE_KEY_A / UCBioAPI_SC_USE_KEY_B 139
- UCBioAPI_SC_LED_TOGGLE / UCBioAPI_SC_LED_NOT_TOGGLE 139

5.2. Error definitions 140

5.2.1. Success 140

- UCBioAPIERROR_NONE 140

5.2.2. General error definitions 141

- UCBioAPIERROR_INVALID_HANDLE 141
- UCBioAPIERROR_INVALID_POINTER 141
- UCBioAPIERROR_INVALID_TYPE 141
- UCBioAPIERROR_FUNCTION_FAIL 141
- UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED 141
- UCBioAPIERROR_ALREADY_PROCESSED 142
- UCBioAPIERROR_EXTRACTION_OPEN_FAIL 142
- UCBioAPIERROR_VERIFICATION_OPEN_FAIL 142
- UCBioAPIERROR_DATA_PROCESS_FAIL 142
- UCBioAPIERROR_MUST_BE_PROCESSED_DATA 142
- UCBioAPIERROR_INTERNAL_CHECKSUM_FAIL 143
- UCBioAPIERROR_ENCRYPTED_DATA_ERROR 143
- UCBioAPIERROR_UNKNOWN_FORMAT 143

- UCBioAPIERROR_UNKNOWN_VERSION 143
- UCBioAPIERROR_VALIDITY_FAIL 143
- UCBioAPIERROR_INVALID_TEMPLATESIZE 144
- UCBioAPIERROR_INVALID_TEMPLATE 144
- UCBioAPIERROR_EXPIRED_VERSION 144
- UCBioAPIERROR_INVALID_SAMPLESPERFINGER 144
- UCBioAPIERROR_UNKNOWN_INPUTFORMAT 145
- UCBioAPIERROR_INVALID_PARAMETER 145
- UCBioAPIERROR_FUNCTION_NOT_SUPPORTED 145

5.2.3. Initialization related error definitions 146
- UCBioAPIERROR_INIT_MAXFINGERSFORENROLL 146
- UCBioAPIERROR_INIT_NECESSARYENROLLNUM 146
- UCBioAPIERROR_INIT_SAMPLESPERFINGER 146
- UCBioAPIERROR_INIT_SECULEVELFORENROLL 146
- UCBioAPIERROR_INIT_SECULEVELFORVERIFY 146
- UCBioAPIERROR_INIT_SECULEVELFORIDENTIFY 146

5.2.4. Device related error definitions 147
- UCBioAPIERROR_DEVICE_OPEN_FAIL 147
- UCBioAPIERROR_INVALID_DEVICE_ID 147
- UCBioAPIERROR_WRONG_DEVICE_ID 147
- UCBioAPIERROR_DEVICE_ALREADY_OPENED 147
- UCBioAPIERROR_DEVICE_NOT_OPENED 147
- UCBioAPIERROR_DEVICE_BRIGHTNESS 148
- UCBioAPIERROR_DEVICE_CONTRAST 148
- UCBioAPIERROR_DEVICE_GAIN 148

5.2.5. User interface related error definitions 149
- UCBioAPIERROR_USER_CANCEL 149
- UCBioAPIERROR_USER_BACK 149
- UCBioAPIERROR_CAPTURE_TIMEOUT 149
- UCBioAPIERROR_CAPTURE_FAKE_SUSPICIOUS 149
- UCBioAPIERROR_ENROLL_EVENT_PLACE 149
- UCBioAPIERROR_ENROLL_EVENT_HOLD 149
- UCBioAPIERROR_ENROLL_EVENT_REMOVE 149
- UCBioAPIERROR_ENROLL_EVENT_PLACE_AGAIN 149
- UCBioAPIERROR_ENROLL_EVENT_PROCESS 149
- UCBioAPIERROR_ENROLL_EVENT_MATCH_FAILED 150

5.2.6. FastSearch related error definitions 151
- UCBioAPIERROR_FASTSEARCH_INIT_FAIL 151
- UCBioAPIERROR_FASTSEARCH_SAVE_DB 151
- UCBioAPIERROR_FASTSEARCH_LOAD_DB 151
- UCBioAPIERROR_FASTSEARCH_UNKNOWN_VER 151

- UCBioAPIERROR_FASTSEARCH_IDENTIFY_FAIL 151
- UCBioAPIERROR_FASTSEARCH_DUPLICATED_ID 152
- UCBioAPIERROR_FASTSEARCH_IDENTIFY_STOP 152
- UCBioAPIERROR_FASTSEARCH_NOUSER_EXIST 152

5.2.7. Optional value related error definitions 153
- UCBioAPIERROR_OPTIONAL_UUID_FAIL 153
- UCBioAPIERROR_OPTIONAL_PIN1_FAIL 153
- UCBioAPIERROR_OPTIONAL_PIN2_FAIL 153
- UCBioAPIERROR_OPTIONAL_SITEID_FAIL 153
- UCBioAPIERROR_OPTIONAL_EXPIRE_DATE_FAIL 153

5.2.8. SmartCard related error definitions 154
- UCBioAPIERROR_SC_FUNCTION_FAILED 154
- UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE 154
- UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE 154

5.3. API References 155

5.3.1. Basic API 155
- UCBioAPI_Init 155
- UCBioAPI_Terminate 156
- UCBioAPI_GetVersion 157
- UCBioAPI_GetInitInfo 158
- UCBioAPI_SetInitInfo 159
- UCBioAPI_SetSkinResource 160
- UCBioAPI_SetCaptureQuality 160

5.3.2. Device related API 161
- UCBioAPI_EnumerateDevice 161
- UCBioAPI_OpenDevice 162
- UCBioAPI_CloseDevice 163
- UCBioAPI_GetDeviceInfo 164
- UCBioAPI_SetDeviceInfo 165
- UCBioAPI_AdjustDevice 166
- UCBioAPI_GetOpenedDeviceID 167
- UCBioAPI_CheckFinger 168
- UCBioAPI_SetAutoDetect 168

5.3.3. Memory related API 169
- UCBioAPI_GetFIRFromHandle 169
- UCBioAPI_GetExtendedFIRFromHandle 170
- UCBioAPI_GetHeaderFromHandle 171
- UCBioAPI_GetExtendedFIRFromHandle 172
- UCBioAPI_GetTextFIRFromHandle 173
- UCBioAPI_GetExtendedTextFIRFromHandle 174
- UCBioAPI_FreeFIRHandle 175

- UCBioAPI_FreeFIR 176
- UCBioAPI_FreeTextFIR 177
- UCBioAPI_FreePayload 178

5.3.4. Core API 179
- UCBioAPI_Capture 179
- UCBioAPI_Process 181
- UCBioAPI_CreateTemplate 182
- UCBioAPI_VerifyMatch 184
- UCBioAPI_VerifyMatchEx 185
- UCBioAPI_Enroll 187
- UCBioAPI_Verify 189

5.3.5. Data conversion API 191
- UCBioAPI_FIRToTemplate 191
- UCBioAPI_TemplateToFIR 193
- UCBioAPI_TemplateToFIREx 195
- UCBioAPI_ConvertTemplateData 197
- UCBioAPI_ImportDataToFIR 199
- UCBioAPI_ImportDataToFIREx 200
- UCBioAPI_AuditFIRToImage 202
- UCBioAPI_ImageToAuditFIR 203
- UCBioAPI_FreeData 204
- UCBioAPI_FreeExportData 205
- UCBioAPI_FreeExportAuditData 206

5.3.6. FastSearch API 207
- UCBioAPI_InitFastSearchEngine 207
- UCBioAPI_TerminateFastSearchEngine 208
- UCBioAPI_GetFastSearchInitInfo 209
- UCBioAPI_SetFastSearchInitInfo 210
- UCBioAPI_AddFIRToFastSearchDB 211
- UCBioAPI_RemoveFpFromFastSearchDB 213
- UCBioAPI_RemoveUserFromFastSearchDB 214
- UCBioAPI_IdentifyFIRFromFastSearchDB 215
- UCBioAPI_ClearFastSearchDB 217
- UCBioAPI_SaveFastSearchDBToFile 218
- UCBioAPI_LoadFastSearchDBToFile 219
- UCBioAPI_GetFpCountFromFastSearchDB 220
- UCBioAPI_GetFpInfoFromFastSearchDB 221
- UCBioAPI_CheckFpExistInFastSearchDB 222

5.3.7. SmartCard API 223
- UCBioAPI_SC_RFPowerOn 223
- UCBioAPI_SC_RFPowerOff 224

- UCBioAPI_SC_RFFunction 225
- UCBioAPI_SC_ReadSerial 226
- UCBioAPI_SC_ReadBlock 227
- UCBioAPI_SC_WriteBlock 229
- UCBioAPI_SC_ReadSector 231
- UCBioAPI_SC_WriteSector 233
- UCBioAPI_SC_ReadSectorFieldContent 235
- UCBioAPI_SC_WriteSectorFieldContent 237
- UCBioAPI_SC_PreValue 239
- UCBioAPI_SC_ReadValue 241
- UCBioAPI_SC_IncrementValue 243
- UCBioAPI_SC_DecrementValue 245
- UCBioAPI_SC_WriteSectorTrailer 247
- UCBioAPI_SC_ReqA 249
- UCBioAPI_SC_WupA 250
- UCBioAPI_SC_Select 251
- UCBioAPI_SC_HaltA 252
- UCBioAPI_SC_Rats 253
- UCBioAPI_SC_PpsRequest 255
- UCBioAPI_SC_BlockFormat 257
- UCBioAPI_SC_Deselect 259
- UCBioAPI_SC_TypeA_ActiveState 260

6. API Reference for COM 261

6.1. UCBioBSP Object 261

6.1.1. Methods 261
- SetSkinResource 261
- SetCaptureQuality 261

6.1.2. Properties 263
- ErrorCode 263
- ErrorDescription 263
- Device 263
- Extraction 263
- Matching 264
- FPData 264
- FPImage 264
- FastSearch 264
- SmartCard 265
- CheckValidityModule 265
- MajorVersion 265

- MinorVersion 265
- BuildNumber 265
- MaxFingersForEnroll 266
- NecessaryEnrollNum 266
- SamplesPerFinger 266
- DefaultTimeout 266
- SecurityLevelForEnroll / SecurityLevelForVerify / SecurityLevelForIdentify 267
- WindowStyle 267
- WindowOption 268
- ParentWnd 269
- FingerWnd 269
- CaptionMsg 269
- CancelMsg 269
- FPForeColor / FPBackColor 270
- DisableFingerForEnroll 270

6.2. IDevice Interface 272

6.2.1. Methods 272
- Open 272
- Close 272
- Enumerate 272
- Adjust 273

6.2.2. Properties 274
- ErrorCode 274
- ErrorDescription 274
- EnumCount 274
- EnumDeviceID 274
- EnumDeviceNameID 275
- EnumInstance 275
- EnumDeviceName 275
- EnumDeviceDescription 276
- EnumDeviceDll / EnumDeviceSys 276
- EnumDeviceAutoOn 276
- EnumDeviceBrightness / EnumDeviceContrast / EnumDeviceGain 277
- OpenedDeviceID 277
- DeviceNameID / DeviceInstance 277
- DeviceID 277
- ImageWidth / ImageHeight 278
- Brightness / Contrast / Gain 278
- IsFingerExisted 278

6.3. IExtraction Interface 280

6.3.1. Methods 280
- Auto Detect 280
- Capture 280
- Enroll 281

6.3.2. Properties 283
- ErrorCode 283
- ErrorDescription 283
- FIR 283
- FIRLength 283
- TextFIR 284
- FIRFormat 284

6.4. IMatching Interface 285

6.4.1. Methods 285
- VerifyMatch 285
- Verify 285

6.4.2. Properties 287
- ErrorCode 287
- ErrorDescription 287
- MatchingResult 287
- IsPayloadExisted 287
- Payload 288
- PayloadLength 288
- TextPayload 288

6.5. IFPData Interface 290

6.5.1. Methods 290
- ClearFPData 290
- Export 290
- Import 292
- CreateTemplate 294

6.5.2. Properties 295
- ErrorCode 295
- ErrorDescription 295
- TotalFingerCount 295
- FingerID 295
- SampleNumber 296
- FPSampleData 296
- FPSampleDataLength 297
- FIR 297
- FIRLength 298
- TextFIR 298

- FIRFormat 299

6.6. IFPImage Interface 300

6.6.1. Methods 300
- Export 300
- ExportEx 300
- Save 301

6.6.2. Properties 303
- ErrorCode 303
- ErrorDescription 303
- TotalFingerCount 303
- FingerID 303
- SampleNumber 304
- ImageWidth / ImageHeight 304
- RawData 305
- AuditData 305
- AuditDataLength 305
- TextAuditData 306

6.7. IFastSearch Interface 307

6.7.1. Methods 307
- AddFIR 307
- RemoveFp 307
- RemoveUser 308
- ClearDB 309
- SaveDBToFile 309
- LoadDBFromFile 309
- IdentifyUser 310

6.7.2. Properties 311
- ErrorCode 311
- ErrorDescription 311
- FpCount 311
- FpInfo 311
- IsFpExisted 312
- AddedFpCount 312
- AddedFpInfo 313
- MatchedFpInfo 313
- MaxSearchTime 313
- UseGroupMatch 314
- MatchMethod 314

6.8. ITemplateInfo Interface 315

6.8.1. Properties 315
- UserID 315
- FingerID 315
- SampleNumber 315

6.9. ISmartCard Interface 316

6.9.1. Methods 316
- RFPowerOn 316
- RFPowerOff 317
- RFFunction 318
- ReadSerial 319
- ReadBlock 320
- WriteBlock 321
- ReadSector 322
- WriteSector 323
- ReadSectorFieldContent 324
- WriteSectorFieldContent 325
- PreValue 326
- ReadValue 327
- IncrementValue 328
- DecrementValue 329
- WriteSectorTrailer 330
- ReqA 331
- WupA 331
- Select 331
- HaltA 332
- Rats 333
- PpsRequest 334
- BlockFormat 335
- Deselect 336
- TypeA_ActiveState 337

6.9.2. Properties 338
- ErrorCode 338
- ErrorDescription 338
- LED 338
- AuthMode 338
- KeyA / KeyB 339
- ResultBuffer 339
- ResultLength 340
- Value 340
- Serial 341

7. Distribution Guide 342

7.1. 공통 사항 342

- 디바이스 드라이버 342
- UCDevice.dll 342
- UCBioBSP.dll 342
- Skin 파일 342

7.2. DLL을 이용해 개발한 경우 342

7.3. COM을 이용해 개발한 경우 343

- UCBioBSPCOM.dll 343

7.4. .NET을 이용해 개발한 경우 343

- .NET Framework v2.0 이상 343
- UNIONCOMM.SDK.UCBioBSP.dll 343

7.5. BioAPI Framework 상에서 개발한 경우 343

- BioAPI Framework v2.0 이상 343
- UCBioBSP.dll 배포 및 등록 343

Appendix A. How to enroll fingerprint properly 344

A.1. 올바른 지문 입력 방법 344

A.2. 잘못된 지문 입력 방법 345

A.3. 인증 실패 시 대처 방안 345

A.4. 지문 등록, 이렇게 하면 편리합니다. 346

A.5. 사용자 지문 상태에 따른 유의 사항 346

1.Overview

UCBioBSP SDK는 UNION COMMUNITY의 지문 인식기를 이용하여 어플리케이션을 개발 할 수 있도록 만든 High-Level SDK로 BioAPI Consortium에서 제시하는 BioAPI v2.0용 SPI를 지원하고 또한 BioAPI에서 사용되는 API와 유사한 형태의 확장된 API를 추가로 제공하는 Software Development Kit 이라 할 수 있다.

UCBioBSP SDK는 지문 인증과 관련된 모든 API들과 그에 따르는 등록 및 인증을 위한 GUI(Graphical User Interface)도 모두 제공하기 때문에 개발자들은 최소한의 노력으로 개발코자 하는 제품에 지문 인식 기능을 추가해 넣을 수 있다.
또한 스마트카드를 위한 API들도 제공되므로 사용자가 원하는 데이터를 스마트카드상에 쓰고 읽을 수 있는 기능도 쉽게 구현 할 수 있다.

이 문서에서는 사용자가 쉽게 사용 할 수 있도록 각 API에 대한 상세한 설명과 그 사용법에 대한 예제를 제공한다.

1.1. SDK의 특징

UCBioBSP SDK는 다음과 같은 특징을 가지고 있다.

- **BioAPI v2.0 호환**
  바이오 인증을 위한 국제 표준 API인 BioAPI v2.0(ISO/IEC 19784-1:2005)을 지원

- **다양한 프로그래밍 언어 제공**
  C, C++, Visual Basic, Delphi, 등등 각종 언어에서 사용 가능한 모듈 제공 및 Sample 제공

- **사용하기 편리한 GUI 제공**
  지문에 최적화된 GUI 제공 및 Skin 방식의 UI 제공으로 사용자에 맞도록 수정 가능

- **Multi fingerprint 지원**
  사용자당 10개의 지문을 하나의 통합된 데이터로 관리 가능

- **강력한 암호화 제공**
  국제 표준 암호화 알고리즘인 AES(Advanced Encryption Standard)를 이용한 데이터 보안

- **자체 보호 기능**
  모듈의 변형 및 위조를 막기 위한 자체 변조 검사 기능 보유

- **스마트카드 지원**
  스마트카드(Mifare, ISO14443-A)에 값을 읽고 쓸 수 있는 기능 제공

1.2. 제공 모듈

UCBioBSP SDK에는 다음과 같은 모듈이 제공된다.

- **UCBioBSP.dll**
  UCBioBSP SDK의 핵심 모듈로서 지문 인증과 관련된 모든 기능을 구현하고 있는 메인 모듈이라 할 수 있다. UCBioBSP SDK를 이용해 개발을 할 경우 이 모듈은 반드시 포함되어야 한다.
  C, C++에서 사용 할 수 있는 API들을 제공하고 있으며 BioAPI v2.0을 위한 SPI도 포함하고 있어 BioAPI Framework 환경에서 등록되어 사용될 수 있다. (BioAPI 관련한 자세한 설명은 http://www.bioapi.org를 참조하거나 BioAPI Framework 제공 업체로부터 얻을 수 있다.)
  관련된 Sample 코드는 제공되는 Samples 폴더의 DLL 폴더와 BioAPI 폴더로 부터 찾을 수 있다.

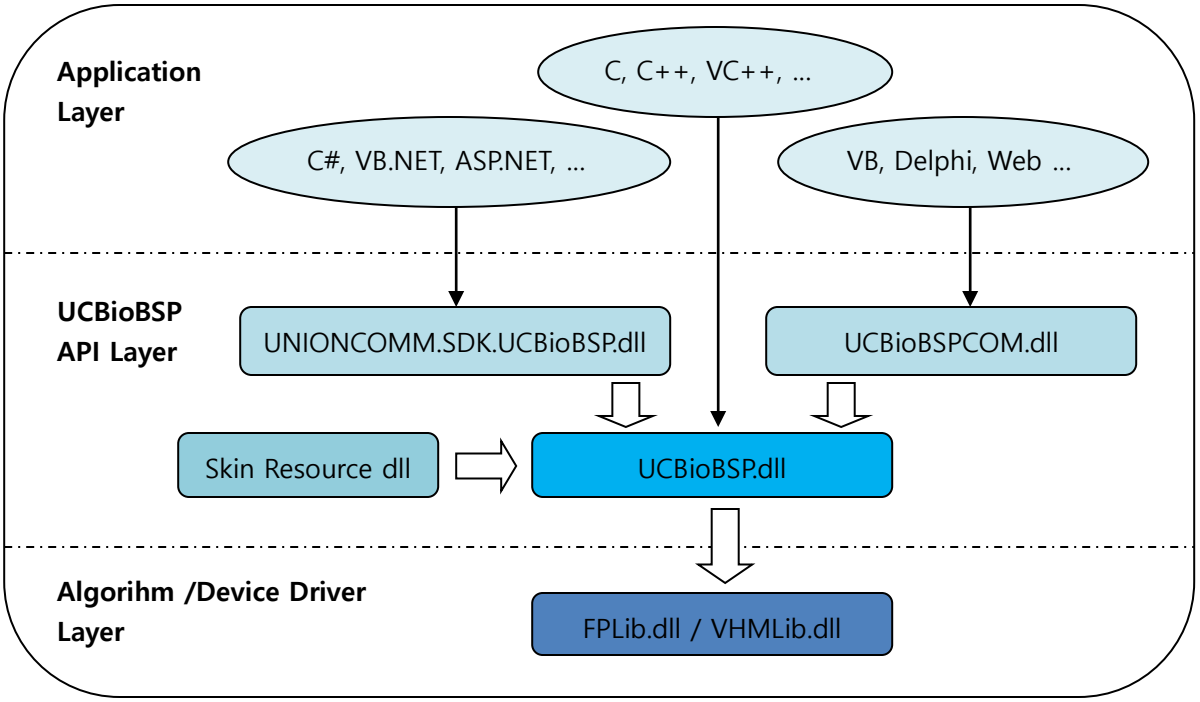
- **UCBioBSPCOM.dll**
  Visual Basic, Delphi 등의 RAD(Rapid Application Development) 툴 사용자와 웹 개발을 지원하기 위한 목적으로 개발된 COM(Component Object Model) 모듈이다.
  UCBioBSPCOM.dll은 UCBioBSP.dll 보다 상위레벨에 존재하기 때문에 UCBioBSP.dll이 반드시 같이 존재해야 동작 할 수 있다. 또한 UCBioBSP.dll에서 제공하는 모든 기능을 제공하지는 않지만 반대로 UCBioBSP.dll에서 제공하지 않는 기능도 일부 가지고 있다.
  관련된 Sample 코드는 제공되는 Samples 폴더의 COM 폴더에서 찾을 수 있다.

- **UNIONCOMM.SDK.UCBioBSP.dll**
  Microsoft의 .NET 환경에서 사용되는 C#, VisualBasic.NET, ASP.NET 등의 언어에서 사용 가능한 .NET용 Class library 모듈이다.
  COM 모듈과 마찬가지로 UNIONCOMM.SDK.UCBioBSP.dll은 UCBioBSP.dll 보다 상위레벨에 존재하기 때문에 UCBioBSP.dll이 반드시 같이 존재해야 동작 할 수 있다.
  관련된 Sample 코드는 제공되는 Samples 폴더의 dotNET 폴더에서 찾을 수 있다.

- **Resource DLL**
  UCBioBSP SDK는 외부에서 만들어진 Skin용 Resource 파일을 사용 할 수 있는 방법을 제공한다. UCBioBSP.dll은 기본적으로 영문 Skin을 내장하고 있는데 만일 다른 언어나 다른 형태의 UI를 사용하고자 한다면 UCBioBSP SDK에서 제공하는 Skin 관련 API를 이용하여 외부에 있는 리소스 파일을 읽어 들일 수 있다.
  현재 SDK에는 기본적으로 영문, 한글 Skin Resource DLL을 제공한다.
  Customized Skin Resource DLL을 만드는 방법은 별도 문서로 제공한다.

1.3. 개발 모델

개발 모델에 대한 구조는 아래와 같다.



1.4. 지문 데이터 구조

UCBioBSP SDK에서 지문 데이터는 FIR(Fingerprint Identification Record) 이라는 데이터 형태로 표현된다. FIR은 한 사람의 지문 데이터를 표현하기 위한 데이터 구조로 하나의 FIR안에 최대 10개 손가락의 지문 데이터를 포함 할 수 있으며 이미지 데이터도 포함 할 수 있는 구조를 가지고 있다.

FIR은 Format, Header, Data Block으로 크게 3부분으로 구성되어 있다.

Format	Header	Data Block
4Bytes	72Bytes	Variable size (=Header. DataLength)

1.4.1. Format

Format은 FIR의 형태를 지정하는 값으로 이 값에 따라 Header의 구조나 FIR의 구조가 달라질 수 있다. 크기는 4bytes이다. 현재는 UCBioAPI_FIR_FORMAT_STANDARD를 의미하는 1의 값만을 가진다. 향후 새로운 FIR 구조가 나올 경우 이 값이 바뀔 수 있다.

1.4.2. Header

Header는 다음과 같은 구조로 되어 있고 크기는 72bytes이다.

Header Length	Data Length	Version	Data Type	Purpose	Quality	Optional Data	Reserved
4Bytes	4Bytes	2Bytes	2Bytes	2Bytes	2Bytes	52Bytes	4Bytes

Length	UUIDInfo	PIN1	PIN2	Privilege	SiteID	IssueDate	ExpireDate	Reserved
4Bytes	20Bytes	4Bytes	4Bytes	4Bytes	4Bytes	4Bytes	4Bytes	4Bytes

- **Header Length**
  Header의 길이 값을 가지고 있다. 이 값은 항상 72의 값을 가진다.

- **Data Length**
  지문 데이터인 Data Block의 길이 값을 가지고 있다. 지문 데이터는 사용자에 따라 달라질 수 있으므로 가변이다.

- **Version**
  FIR의 버전 정보를 담고 있다. 현재는 1의 값을 가진다.

- **Data Type**
  FIR에 저장되어 있는 지문 데이터의 형태를 가리킨다. FIR은 Raw 이미지 데이터와 지문의 특징점 데이터를 가질 수 있으며 데이터의 암호화 여부도 가리킨다.

- **Purpose**
  FIR 데이터가 사용되어질 목적을 나타낸다. Verify, Identify, Enroll, Audit, ... 등의 목적에 따라 다른 값을 가지게 된다.

- **Quality**
  지문 데이터의 품질값을 가진다. 0에서 100까지의 값을 가질 수 있으며 숫자가 클수록 좋은 품질의 지문데이터이다.

- **Optional Data**
  FIR에 추가적인 정보를 기록 할 수 있는 공간이다. 향후 지문 인증에서 좀 더 세밀한 인증을 하고자 할 경우 이 값들을 사용 할 수 있다. 가질 수 있는 정보는 아래와 같다.

  ■ **Length**
  Optional Data의 길이 값을 가지고 있다. 이 값은 항상 52의 값을 가진다.

  ■ **UUIDInfo**
  FIR의 고유 UUID(Universally Unique Identifier) 값에 대한 정보를 지정 할 수 있다.
  UCBioAPI_FIR_UUID_INFO 구조체의 값을 가진다.

  ■ **PIN1/PIN2**
  PIN(Personal Identification Number) 정보를 저장 할 수 있다.

  ■ **Privilege**
  FIR의 권한 정보를 지정 할 수 있다. 1에서 9까지 값을 가질 수 있으며 숫자가 클수록 높은 권한이 요구된다. 0일 경우 사용되지 않는다.

  ■ **Site ID**
  FIR 데이터가 사용되어야 할 특정 Site의 ID를 지정 할 수 있다.

  ■ **Issue Date**
  FIR 데이터가 만들어진 날짜를 지정 할 수 있다.

  ■ **Expire Date**
  FIR 데이터가 언제까지 유효한지를 지정 할 수 있다.

  ■ **Reserved**
  예약된 영역

- **Reserved**
  예약된 영역

1.4.3. Data Block

지문 데이터가 저장된 영역이다. Binary 형태의 암호화된 영역이며 지문의 특징점 데이터를 가지거나 또는 지문의 Raw 이미지 데이터를 가진다.
사용자에 따라 지문의 크기와 등록된 손가락의 개수에 따라 Data Block의 크기는 달라질 수 있다. Data Block의 크기는 Header의 Data Length 값을 참조하여 알 수 있다.

1.5. 용어 설명

이 문서에서 사용 되어질 용어들에 대한 설명이다.

- **BSP**
  Biometric Service Provider의 약자로 바이오 인증을 위한 서비스 제공 모듈을 의미한다.

- **템플릿(Template) / 샘플(Sample)**
  하나의 지문에 대한 지문 특징점 데이터를 의미한다.

- **Payload**
  FIR 데이터 내부에 사용자의 특정 정보를 담을 수 있는 영역을 의미한다. 이렇게 담아둔 Payload 정보는 암호화 된 상태로 FIR 내부에 저장되어 있다가 인증이 성공 할 경우에만 그 값을 다시 얻어올 수 있다.

- **FIR**
  Fingerprint Identification Record의 약자로 사용자별 지문 데이터 값이다. 하나의 FIR안에는 다수의 손가락에 대한 Template 정보 또는 Raw 이미지 정보와 Payload 정보 등을 담고 있을 수 있다.

- **FastSearch**
  UCBioBSP SDK에서 제공하는 1:N 인증을 위한 고속 인증 엔진의 이름이다.

- **BioAPI**
  BioAPI Consortium에서 지정한 바이오 인증을 위한 Application Programming Interface의 집합을 국제 표준(ISO/IEC 19784-1:2005)으로 지정한 것이다. 자세한 설명은 http://www.bioapi.org 를 통해 얻을 수 있다.

2.Installation

2.1. 시스템 요구 사항

UCBioBSP SDK를 설치하기 위해서는 다음과 같은 시스템 요구사항이 필요하다.

- **OS**
  Windows 98SE/ME/2000/XP/2003/Vista, 그 외 USB가 지원되는 Windows OS 모두 지원.

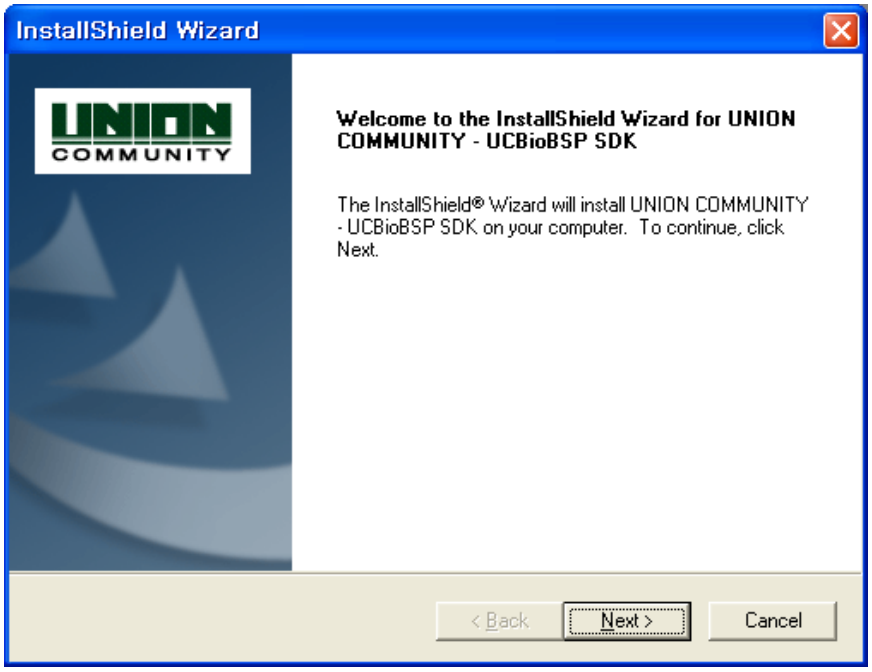
- **CPU**
  Pentium 이상의 CPU

- **Web 환경**
  Server: IIS(Internet Information Server) 4.0 이상
  Client: IE(Internet Explorer) 5.0 이상

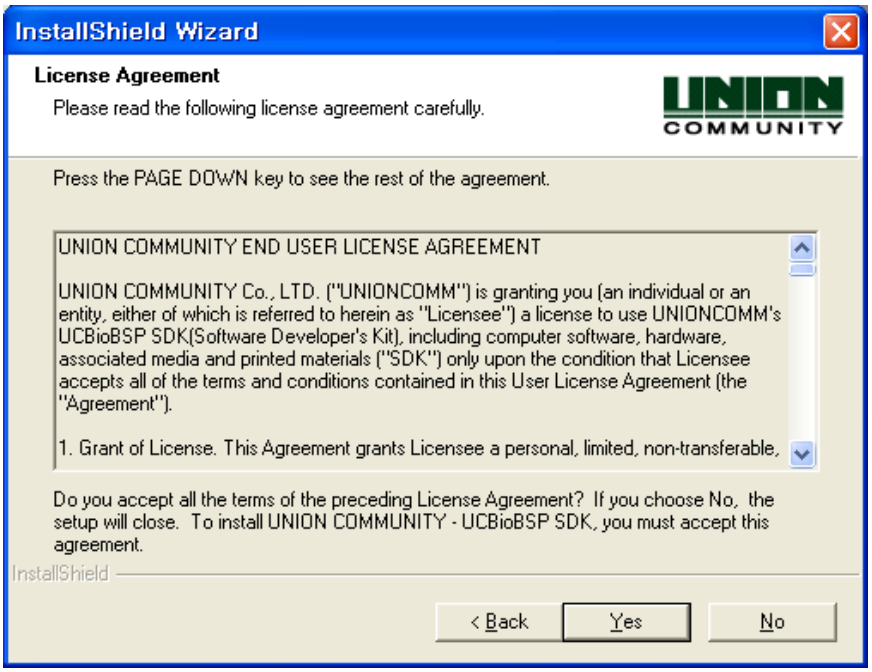
- **Device**
  지문 획득을 위한 UNION COMMUNITY의 USB 지문 인식기
  UCBioBSP SDK는 현재 UNION COMMUNITY에서 나온 모든 PC 주변기기용 지문 인식기를 지원하며 향후 출시될 디바이스에 대해서는 개발된 소스의 변경 없이 지원 가능하도록 SDK를 제공 할 예정이다.
  UCBioBSP SDK는 디바이스 드라이버는 포함하고 있지 않으므로 각 디바이스에 대한 디바이스 드라이브는 디바이스 별로 따로 설치를 해 주어야 한다.

2.2. 설치하기

1) 설치 CD를 드라이브에 넣는다.
2) 설치 CD에 있는 Setup.exe 파일을 수행한다.
3) Next 버튼을 눌러 설치를 시작한다.



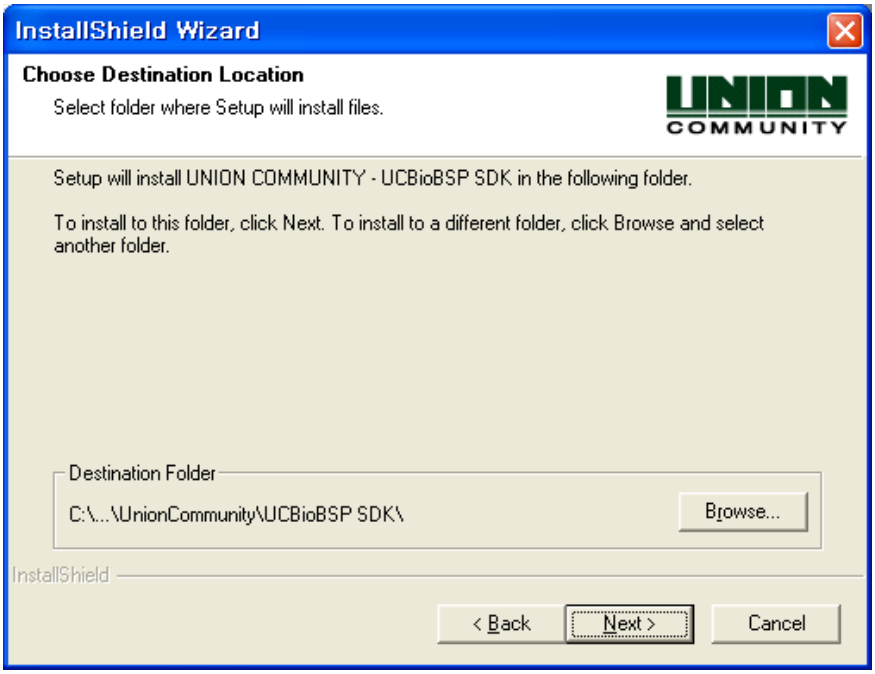
4) 라이선스를 확인 후 라이선스에 동의 할 경우 Yes 버튼을 누른다.



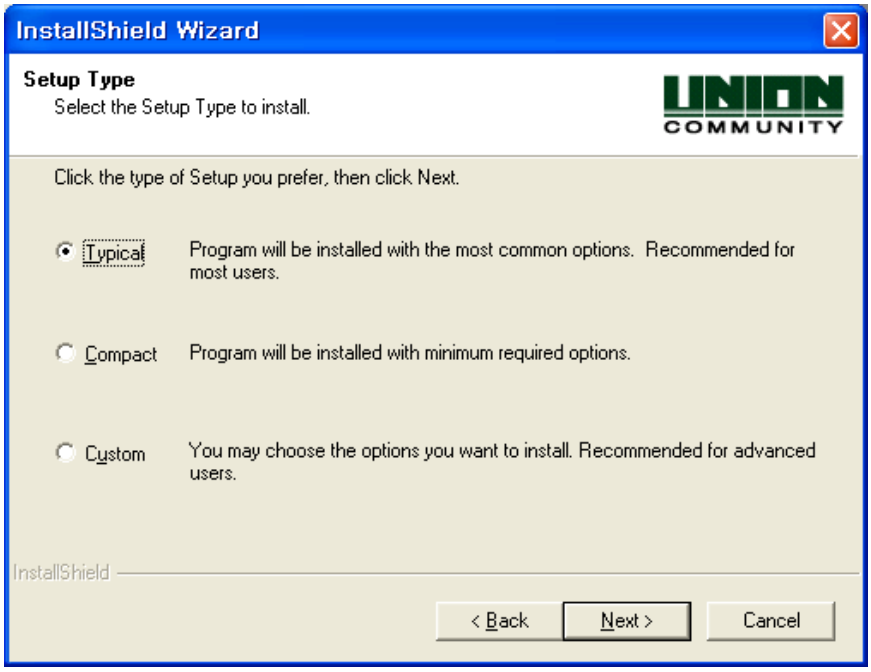
5) 사용자명, 회사명, 그리고 Serial Number를 입력한다. Serial Number는 SDK 구입 시 얻을 수 있다.



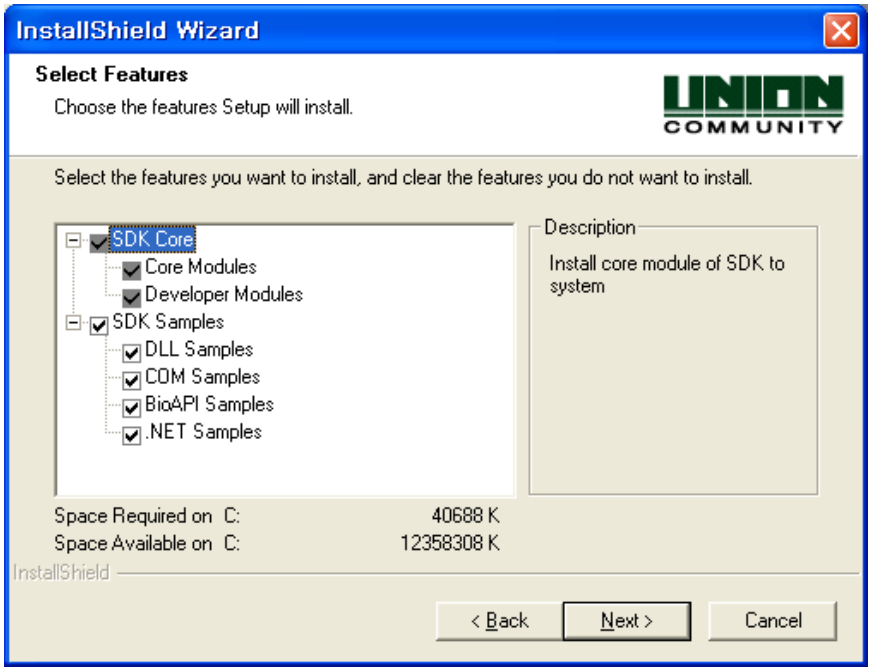
6) 설치 폴더를 지정한다.



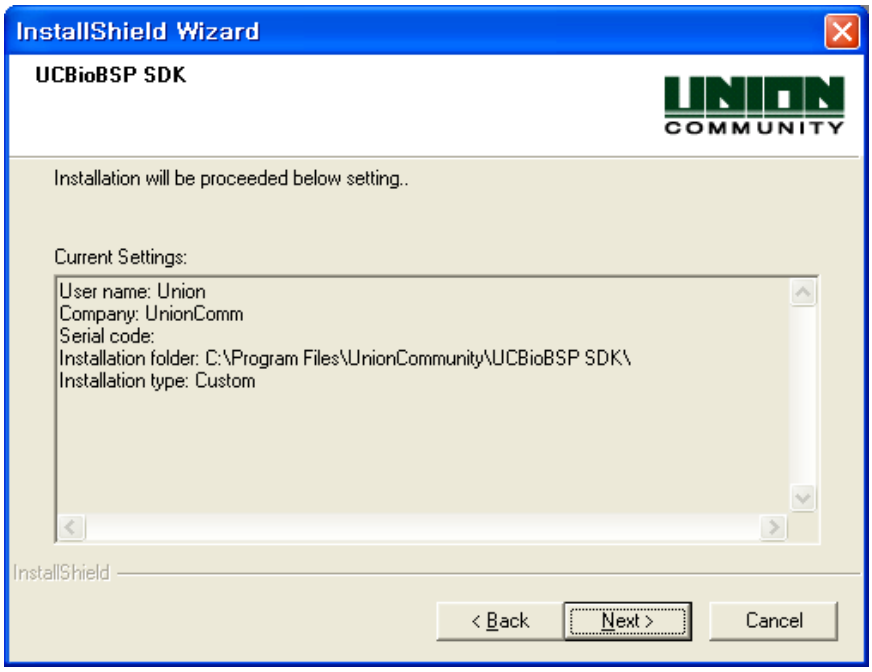
7) 설치 Type을 지정한다. Typical을 선택 할 경우 일반적인 설치로 SDK의 모든 부분을 설치하게 되며, Compact를 선택 할 경우 SDK의 Sample code를 제외하고 설치한다. 만약 설치 구성요소를 직접 선택하고자 할 경우에는 Custom을 선택하여 설치하면 된다.



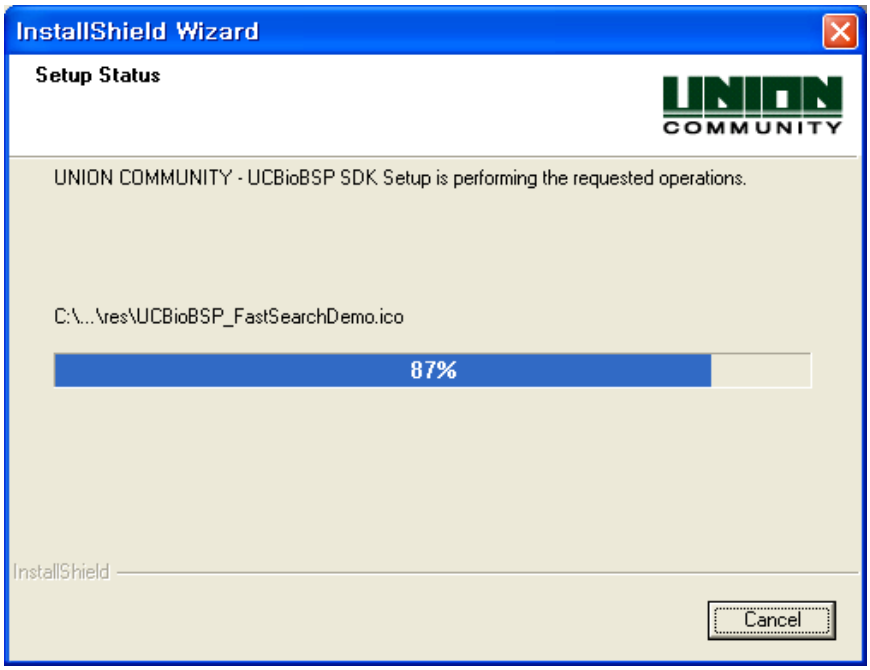
8) 만약 Custom을 선택 한 경우에는 다음과 같은 설치 구성요소를 선택해 설치 할 수 있다.



9) 설치 옵션을 모두 확인 한 후 Next 버튼을 누르면 설치가 진행된다.



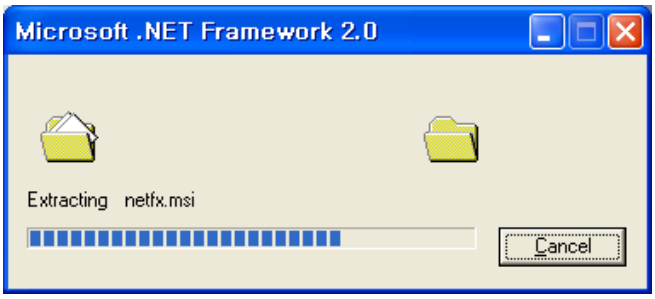
10) SDK의 설치가 진행된다.



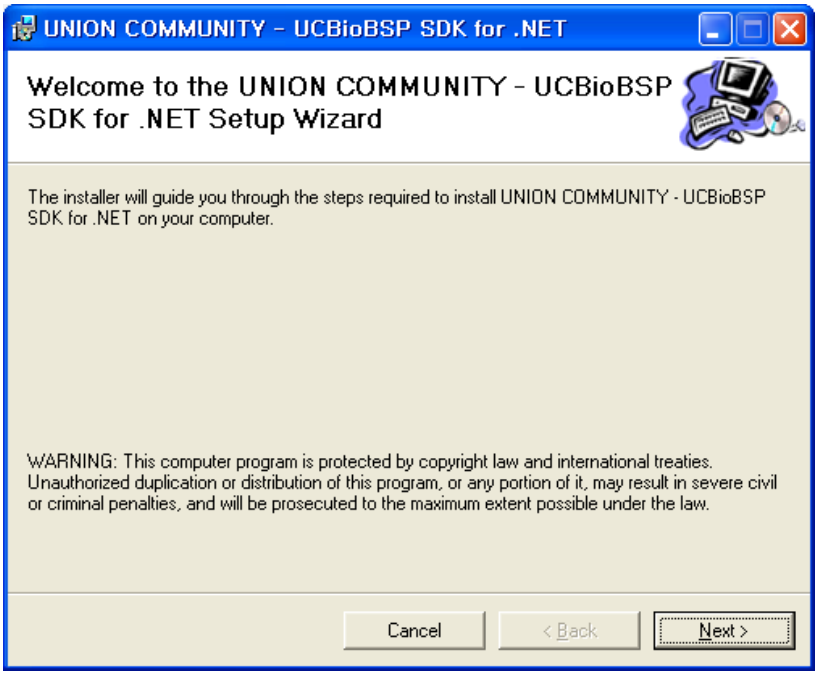
11) 파일 복사가 모두 끝나면 .NET Framework용 모듈을 추가로 설치 할 것인지를 묻는다. 만약 .NET Framework을 사용하지 않는다면 굳이 .NET용 Class library를 설치 할 필요가 없으므로 "아니오"를 선택하면 된다. 향후 필요 시 .NET용 모듈은 따로도 설치가 가능하다. 하지만 .NET Framework 상에서 개발을 해야 할 필요가 있다면 "예"를 선택해 설치를 진행한다.



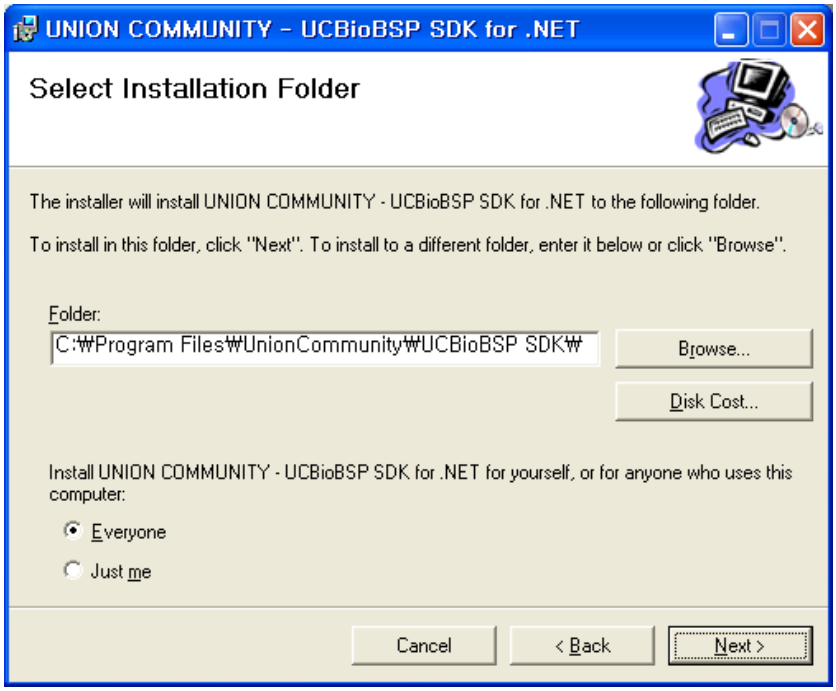
12) "예"를 선택할 경우 다음과 같이 우선 .NET Framework 2.0이 설치가 된다. 만약 시스템이 이미 .NET Framework이 설치되어 있다면 이 작업을 취소하면 된다.



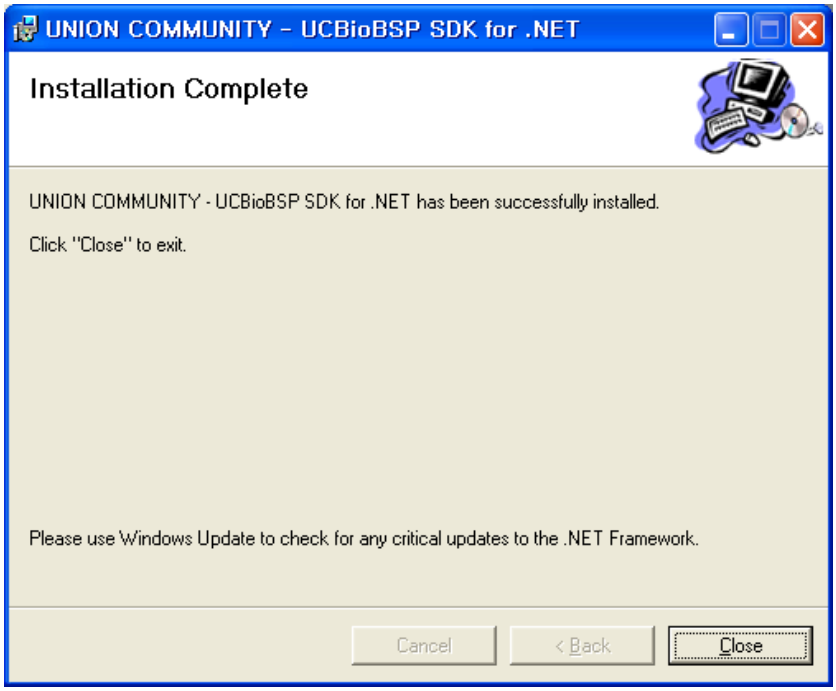
13) .NET Framework의 설치가 끝나면 .NET용 Class library의 설치가 진행된다. Next 버튼을 눌러 설치를 시작한다.



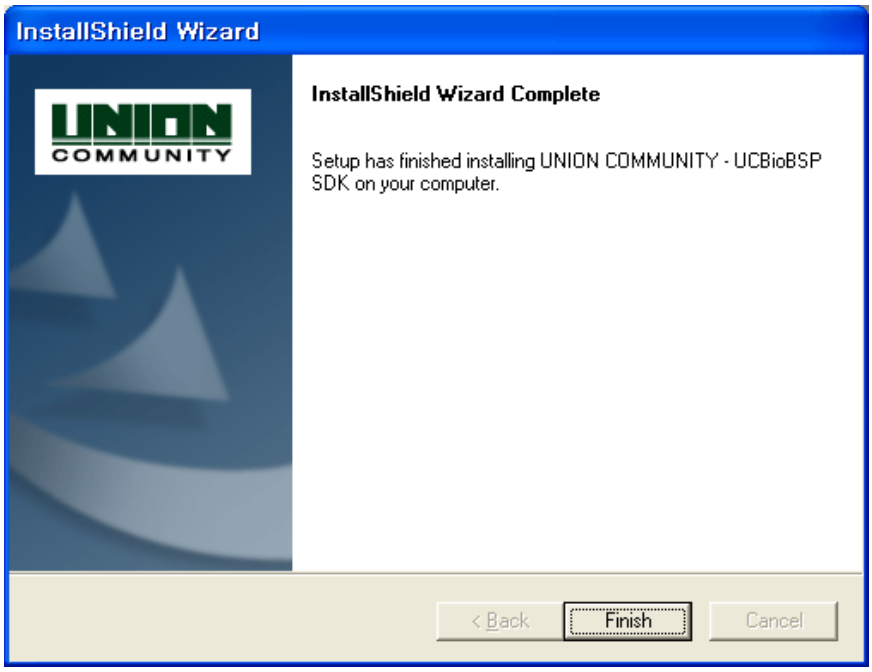
14) 설치 할 폴더를 지정한다. 지정된 폴더 하위의 Bin 폴더에 .NET용 Class library가 설치되고 자동으로 GAC(Global Assembly Cache)에도 등록이 된다. Next 버튼을 눌러 설치를 진행한다.



15) .NET용 모듈의 설치가 완료되면 Close 버튼을 눌러 .NET용 설치프로그램을 종료한다.



16) 다시 원래 SDK의 설치화면으로 돌아와 모든 설치를 마치게 된다.



2.3. 설치되는 파일 및 폴더 설명

설치가 끝나면 다음과 같은 파일들이 시스템에 설치되게 된다.

2.3.1. Windows System32 폴더

SDK의 사용을 위한 Core 모듈이 설치된다. 아래의 파일들이 설치된다.

■ **UCBioBSP.dll**
UCBioBSP SDK의 핵심 모듈. SDK의 모든 기능의 수행을 담당한다.
BioAPI용 SPI(Service Provider Interface)도 제공하므로 BioAPI Framework에 등록될 수 있다.

■ **UCBioBSPCOM.dll**
UCBioBSP SDK의 COM용 모듈.

2.3.2. GAC(Global Assembly Cache) 폴더

.NET Framework 환경을 위한 Class Library가 설치되는 GAC폴더에 아래의 파일이 설치된다.
SDK의 설치 중에 .NET용 라이브러리를 설치할 경우 설치된다.

■ **UNIONCOMM.SDK.UCBioBSP.dll**
.NET용 Class Library 모듈.

2.3.3. (설치폴더)\Inc

SDK를 이용해 C, C++ 언어에서 개발하기 위해 필요한 각종 Header 파일들이 들어있다.

■ **UCBioAPI.h**
UCBioBSP SDK의 메인 Header 파일로 이 파일을 include할 경우 내부적으로 UCBioAPI_Basic.h, UCBioAPI_Error.h, UCBioAPI_Type.h 파일이 자동으로 포함된다.

■ **UCBioAPI_Basic.h**
UCBioBSP SDK에서 사용되는 기본 데이터 타입을 정의하고 있다.

■ **UCBioAPI_Error.h**
UCBioBSP SDK에서 사용되는 에러 값을 정의하고 있다.

■ **UCBioAPI_Type.h**
UCBioBSP SDK에서 사용되는 데이터 타입과 구조체 정보 등을 정의하고 있다.

■ **UCBioAPI_Export.h**
FIR 데이터의 변환을 위한 함수를 정의하고 있다.

■ **UCBioAPI_ExportType.h**
FIR 데이터의 변환을 위한 데이터 타입과 구조체 정보 등을 정의하고 있다.

■ **UCBioAPI_FastSearch,h**
1:N 매칭을 위한 검색엔진을 사용하기 위한 함수를 정의하고 있다.

■ **UCBioAPI_FastSearchType.h**
1:N 매칭을 위한 검색엔진을 사용하기 위한 데이터 타입과 구조체 정보 등을 정의하고 있다.

■ **UCBioAPI_SmartCard,h**
스마트카드를 사용하기 위한 함수를 정의하고 있다.

■ **UCBioAPI_SmartCardType.h**
스마트카드를 사용하기 위한 데이터 타입과 구조체 정보 등을 정의하고 있다.

2.3.4. (설치폴더)\Lib
SDK를 이용해 VC++에서 개발하기 위한 Link용 Library 파일이 들어있다.

■ **UCBioBSP.lib**
VC++용으로 만들어진 Link용 Library 파일. VC++에서 UCBioBSP.dll을 정적으로 Link 할 때 사용된다.

2.3.5. (설치폴더)\Bin
SDK의 수행에 필요한 Core 파일 및 Sample용 실행 파일들이 들어있다.

■ **UCBioBSP.dll / UCBioBSPCOM.dll**
Windows system32 폴더에 설치된 파일과 동일한 파일.

■ **UCBioBSPCOM.cab**
UCBioBSPCOM.dll을 Web에서 배포가 가능하게 압축해 Sign을 한 CAB 파일.

■ **Demo application**
UCBioBSP SDK의 기능을 간단히 테스트 해 볼 수 있는 다수의 Demo 프로그램들이 들어있다. Demo 프로그램의 소스는 Samples 폴더에서 모두 제공된다.

2.3.6. (설치폴더)\dotNET
SDK의 수행에 필요한 dotNET용 Class Library 파일이 들어있다.

■ **UNIONCOMM.SDK.UCBioBSP.dll**
.NET용 Class Library 모듈. GAC에 설치되는 파일과 동일한 파일.

2.3.7. (설치폴더)\dotNet\Setup

.NET용 Class Library를 GAC에 설치하기 위한 설치파일이 들어있다.

■ **Setup.exe (UCBioBSP.NET_Setup.msi)**
.NET용 Class Library 설치 파일

2.3.8. (설치폴더)\Samples

각 언어별 Sample source code가 폴더별로 구분되어 들어있다.

■ **BioAPI**
BioAPI Framework에서 수행되는 BioAPI용 Sample code가 들어있다.
단, 이 Sample이 수행되기 위해서는 반드시 BioAPI Framework v2.0이 설치되어 있어야 하며 UCBioBSP.dll이 Framework에 등록되어 있어야 한다.

■ **COM**
UCBioBSPCOM.dll을 이용해 개발 가능한 Sample code가 들어있다.
1) VB6: Visual Basic 6.0용으로 만들어진 Sample이 들어있다.
2) ASP: IIS에서 수행되는 ASP(Active Server Page)용으로 만들어진 Sample이 들어있다.

■ **DLL**
UCBioBSP.dll을 이용해 개발 가능한 Sample code가 들어있다.
1) VC6: Visual C++ 6.0용으로 만들어진 Sample이 들어있다.

■ **dotNET**
UNIONCOMM.SDK.UCBioBSP.dll을 이용해 Microsoft의 .NET 환경에서 개발 가능한 Sample code가 들어있다.
1) C#: VisualStudio.NET 2005, C#용으로 만들어진 Sample이 들어있다.

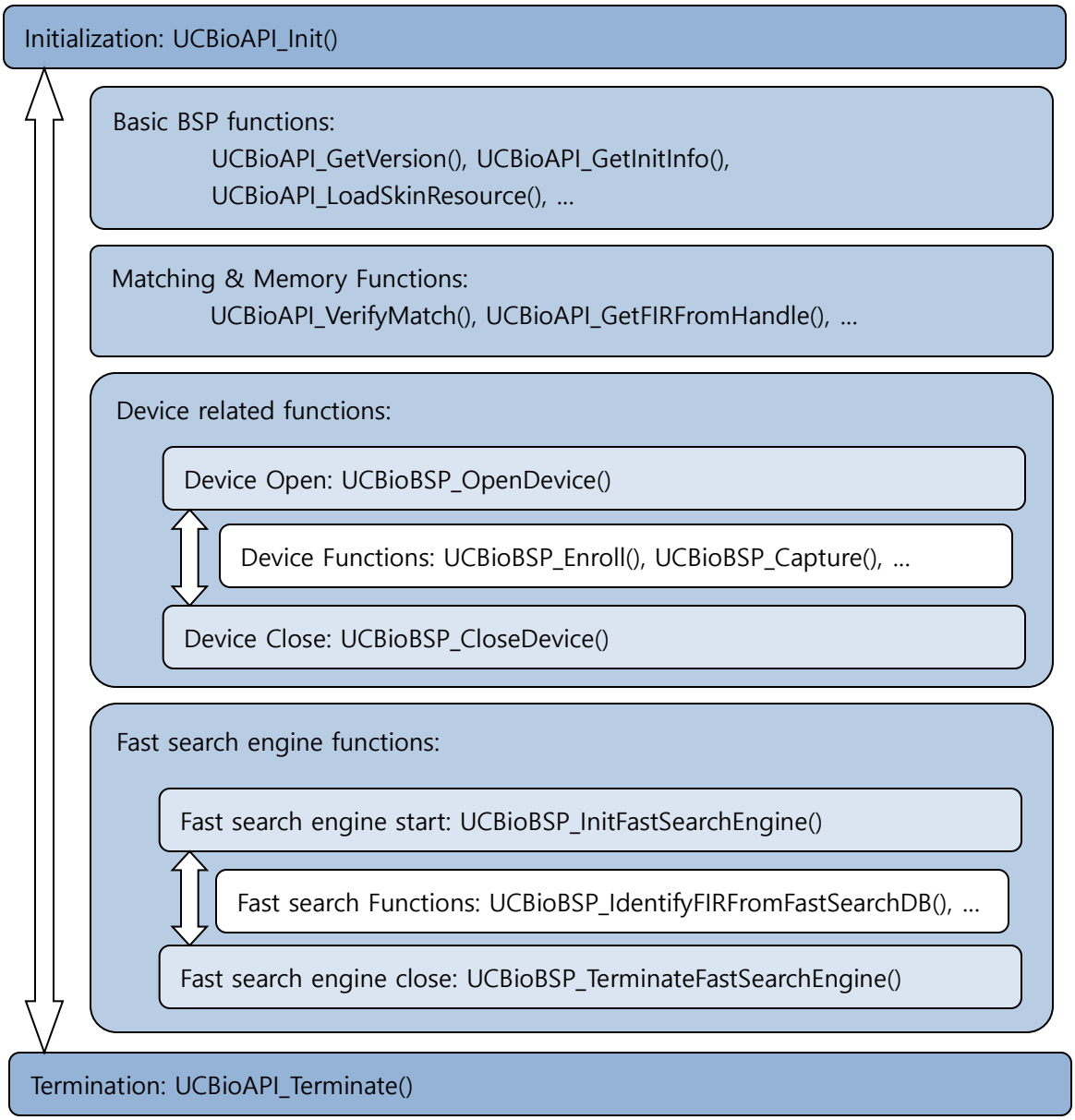
2.3.9. (설치폴더)\Skins

각 언어별 Skin resource 파일이 포함되어 있다. 현재는 영문과 한글만 포함되어 있음.

3.Programming by using DLL

이 장에서는 DLL을 이용해 프로그래밍 하는 방법에 대해 설명한다.
사용언어는 Visual C++ 언어를 기준으로 설명하므로 다른 언어 사용자는 각 언어에 맞도록 적절히 변형하여 사용하여야 한다.

3.1. 함수 호출 구조



3.2. 초기화 및 종료

UCBioBSP SDK를 초기화 하고 종료하는 법에 대한 설명을 한다.

3.2.1. 초기화 하기

UCBioBSP SDK를 사용하기 위해서는 가장 먼저 UCBioAPI_Init 함수를 불러 초기화를 해 주어야 한다. UCBioAPI_Init 함수는 UCBioBSP SDK의 Handle 값을 리턴해 주는데 이 Handle값은 SDK에서 제공하는 거의 모든 함수에서 사용된다.

■ **Example**

```
UCBioAPI_HANDLE m_hUCBioAPI;
…
UCBioAPI_RETURN err = UCBioAPI_Init(&m_hUCBioAPI);
if (err != UCBioAPIERROR_NONE) {
   // Failed to initialize UCBioBSP
} else {
   // Succeeded to initialize UCBioBSP
}
```

3.2.2. 종료 하기

SDK의 사용을 종료하려면 반드시 UCBioAPI_Terminate 함수를 호출하여 종료해 주어야 한다. 이렇게 함으로서 UCBioBSP SDK 내부에서 사용되고 있던 메모리를 모두 해제 할 수 있다.

■ **Example**

```
UCBioAPI_HANDLE m_hUCBioAPI;
…
UCBioAPI_RETURN err = UCBioAPI_Terminate(m_hUCBioAPI);
if (err != UCBioAPIERROR_NONE) {
   // Failed to terminate UCBioBSP
} else {
   // Succeeded to terminate UCBioBSP
}
```

3.3. 기본 설정

UCBioBSP SDK를 초기화가 성공하고 나면 SDK를 사용하기 전에 기본적인 설정들을 얻어오거나 새로운 값으로 지정 할 수 있다.

3.3.1. SDK 버전 얻기

현재 사용중인 SDK의 BSP 버전을 얻어 올 수 있다. 버전은 4Step으로 구성되어 있음

■ **Example**

```
UCBioAPI_VERSION  ver;
memset(&ver, 0, sizeof(UCBioAPI_VERSION));

if (UCBioAPI_GetVersion(m_hUCBioAPI, &ver) == UCBioAPIERROR_NONE) {
   CString szVer;
   szVer.Format(_T("UCBioBSP Version : v%d.%d.%d.%d"),
      ver.Major1, ver.Major2, ver.Minor1, ver.Minor2);
   SetWindowText(szVer);
} else {
   // Failed to get version of BSP
}
```

3.3.2. 기본 설정 값 얻어오기 및 새로운 값 설정

SDK의 기본 설정 값을 얻어오거나 새로운 값을 설정 할 수 있다.

■ **Example**

```
UCBioAPI_INIT_INFO_0 initInfo0;
memset(&initInfo0, 0, sizeof(UCBioAPI_INIT_INFO_0));

UCBioAPI_RETURN err = UCBioAPI_GetInitInfo(m_hUCBioAPI, 0, &initInfo0);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to get init information
} else {
   // Failed to get init information
}
```

■ **Example**

```
UCBioAPI_INIT_INFO_0 initInfo0;
memset(&initInfo0, 0, sizeof(UCBioAPI_INIT_INFO_0));

initInfo0.StructureType = 0;
```

```
initInfo0.MaxFingersForEnroll = m_nMaxFingersForEnroll;
initInfo0.NecessaryEnrollNum = m_nNecessaryEnrollNum;
initInfo0.SamplesPerFinger = m_nSamplesPerFinger;
initInfo0.DefaultTimeout = m_nDefaultTimeout;
initInfo0.SecurityLevelForEnroll = m_nSecuLevelForEnroll;
initInfo0.SecurityLevelForVerify = m_nSecuLevelForVerify;
initInfo0.SecurityLevelForIdentify = m_nSecuLevelForIdentify;

UCBioAPI_RETURN err = UCBioAPI_SetInitInfo(m_hUCBioAPI, 0, &initInfo0);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to set init information
} else {
   // Failed to set init information
}
```

UCBioAPI_INIT_INFO_0 구조체는 다음과 같은 값들을 나타낸다.

■ **StructureType**
반드시 0의 값을 가진다.

■ **MaxFignersForEnroll**
지문 등록을 할 경우 최대 등록 가능하게 할 손가락의 개수를 지정한다.
예를 들어 이 값이 2로 지정될 경우 UCBioAPI_Enroll 함수를 이용해 지문을 등록 할 경우 최대 2개의 손가락만 등록이 가능하다.
디폴트로 이 값은 10을 가진다.

■ **NecessaryEnrollNum**
지문 등록을 할 경우 최소 등록해야 할 손가락의 개수를 지정한다.
이 값은 반드시 MaxFingersForEnroll 값보다는 작거나 같아야 한다.
예를 들어 이 값이 2로 지정될 경우 UCBioAPI_Enroll 함수를 이용해 지문을 등록 할 경우 최소 2개의 손가락이 등록되어야지만 등록 과정의 종료가 가능하다.
디폴트로 이 값은 1을 가진다.

■ **SamplesPerFinger**
지문 등록 시 손가락당 몇 개의 Sample이 저장되는지를 지정한다. 현재는 2로 고정되어 있으며 수정 할 수 없다.

■ **DefaultTimeout**
지문 인증 및 등록 시 지문을 획득하기 위해 디바이스가 동작하는 기본 최대 시간을 ms단위로 지정한다. Timeout은 향후 함수 호출 시 따로 지정 할 수 있으나 이때 UCBioAPI_USE_DEFAULT_TIMEOUT(-1)을 지정한 경우 이 값이 사용된다.

디폴트로 이 값은 10000(10초)을 가진다.

■ **SecurityLevelForEnroll / SecurityLevelForVerify / SecurityLevelForIdentify**
지문 등록 / 인증 / 1:N 인증을 위해 사용되어질 인증 보안 레벨을 각각에 대해 설정 할 수 있다. 이 값은 아래와 같은 값을 가질 수 있다.

```
#define UCBioAPI_FIR_SECURITY_LEVEL_LOWEST          (1)
#define UCBioAPI_FIR_SECURITY_LEVEL_LOWER           (2)
#define UCBioAPI_FIR_SECURITY_LEVEL_LOW             (3)
#define UCBioAPI_FIR_SECURITY_LEVEL_BELOW_NORMAL    (4)
#define UCBioAPI_FIR_SECURITY_LEVEL_NORMAL          (5)
#define UCBioAPI_FIR_SECURITY_LEVEL_ABOVE_NORMAL    (6)
#define UCBioAPI_FIR_SECURITY_LEVEL_HIGH            (7)
#define UCBioAPI_FIR_SECURITY_LEVEL_HIGHER          (8)
#define UCBioAPI_FIR_SECURITY_LEVEL_HIGHEST         (9)
```

기본값으로 Enroll/Verify는 5의 값을 가지고 Identify는 6의 값을 가진다.

■ **Reserved1 / Reserved2**
예약 영역으로 사용하지 않는다. 0의 값을 가진다.

3.3.3. 입력 지문에 대한 품질 설정

입력 지문에 대한 품질(지문 이미지의 품질)을 사용 용도에 맞춰 사전에 SDK에 설정할 수 있다
각 항목별 설정값은 1부터 100까지이며 0으로 설정할 경우 해당 항목은 반영되지 않고 이전값을 그대로 유지한다.
미설정시 SDK는 각 항목별 기본값인 50으로 설정되어 동작하게 된다.

■ **Example**

```
UCBioAPI_SetCaptureQuality(m_hUCBioAPI, 0, 50, 0, 0);
```

3.4. 디바이스 사용하기

지문 등록 및 획득을 위해서는 지문 인식 디바이스를 먼저 사용 가능하게 여는 과정이 필요하다. 여기에서는 PC에 설치된 디바이스를 사용하기 위해 필요한 함수들에 대해 설명한다.

3.4.1. 디바이스 목록 얻기

디바이스를 사용하기 전 UCBioAPI_EnumerateDevice 함수를 사용하여 현재 PC에 연결되어 있는 디바이스의 전체 목록 및 개수 등의 정보를 얻어 올 수 있다.

■ **Example**

```
UCBioAPI_UINT32            m_nNumDevice;
UCBioAPI_DEVICE_ID*        m_pDeviceID;
UCBioAPI_DEVICE_INFO_EX*   m_pDeviceInfoEx;
…

int nIndex;
UCBioAPI_RETURN err = UCBioAPI_EnumerateDevice(m_hUCBioAPI,
                          &m_nNumDevice, &m_pDeviceID,
                          &m_pDeviceInfoEx);
if (err) {
   // Failed to enumerate device!
   return;
}

if (m_nNumDevice == 0) {   // No device
    nIndex = m_comboDeviceList.AddString(_T("--NO DEVICE--"));
    m_comboDeviceList.SetItemData(nIndex, UCBioAPI_DEVICE_ID_NONE);
    return;
}

for (UCBioAPI_UINT32 i = 0 ; i < m_nNumDevice; i++) {
   nIndex = m_comboDeviceList.AddString(m_pDeviceInfoEx[i].Name);
   m_comboDeviceList.SetItemData(nIndex, m_pDeviceInfoEx[i].NameID);
}
```

UCBioAPI_EnumerateDevice 함수를 사용하면 두 번째 인자인 m_nNumDevice에는 시스템에 연결되어 있는 디바이스의 전체 개수가 담겨오고, 세 번째 인자인 m_pDeviceID에는 Device ID가 배열에 담겨서 넘어오게 되며 네 번째 인자인 m_pDeviceInfoEx에는 디바이스에 대한 좀 더 세밀한 정보가 배열에 담겨서 넘어오게 된다.
m_pDeviceID와 m_pDeviceInfoEx는 각각 UCBioAPI_DEVICE_ID와 UCBioAPI_DEVICE_INFO_EX 구조체에 대한 포인터 배열로서 UCBioBSP SDK 내부적으로 메모리 관리를 하므로 Application에서

메모리를 할당하거나 해제를 할 필요가 없다.

3.4.2. 디바이스 열기

디바이스를 사용하기 전에는 UCBioAPI_OpenDevice 함수를 이용해 반드시 사용할 디바이스를 Open해 주어야 한다. 특정 디바이스를 Open하기 위해서는 인자로 Open하고자 하는 디바이스의 ID를 지정해 주면 된다. 만약 최근에 사용한 디바이스를 자동으로 Open하고자 할 경우에는 디바이스 ID에 UCBioAPI_DEVICE_ID_AUTO를 지정하면 된다.

■ **Example**

```
int nDeviceID = UCBioAPI_DEVICE_ID_AUTO;
UCBioAPI_RETURN err = UCBioAPI_OpenDevice(m_hUCBioAPI, nDeviceID);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to open device
} else {
   // Failed to open device
}
```

디바이스 ID로 사용 할 수 있는 값들은 아래와 같다.

```
#define UCBioAPI_DEVICE_NAME_FOH01            (0x01)
#define UCBioAPI_DEVICE_NAME_FOM01            (0x02)
#define UCBioAPI_DEVICE_NAME_FOH03            (0x03)
#define UCBioAPI_DEVICE_NAME_HAM500           (0x04)
#define UCBioAPI_DEVICE_NAME_FOH01A           (0x05)
#define UCBioAPI_DEVICE_NAME_FOM01A           (0x06)
#define UCBioAPI_DEVICE_NAME_FPR02            (0x07)
#define UCBioAPI_DEVICE_NAME_FSH01RF          (0x08)
#define UCBioAPI_DEVICE_NAME_FOH01RF          (0x09)
#define UCBioAPI_DEVICE_NAME_FR100            (0x0a) // 10
#define UCBioAPI_DEVICE_NAME_FPR02LFD         (0x0b) // 11
#define UCBioAPI_DEVICE_NAME_FOH01RFL         (0x0c) // 12
#define UCBioAPI_DEVICE_NAME_FSH01SC          (0x0d) // 13
#define UCBioAPI_DEVICE_NAME_FPR02_V30        (0x0e) // 14
#define UCBioAPI_DEVICE_ID_AUTO               (0x00ff) // 255
```

3.4.3. 디바이스 닫기

디바이스의 사용이 모두 끝난 뒤에는 UCBioAPI_CloseDevice 함수를 이용해 디바이스를 사용 종료를 해 주어야 한다. 이때에는 반드시 처음에 Open한 디바이스를 닫아주어야 한다.
또한 다른 디바이스를 Open하기 위해서는 현재 사용하고 있는 디바이스를 우선 닫아주어야 새

로운 디바이스를 Open 할 수 있다.

■ **Example**

```
int nDeviceID = UCBioAPI_DEVICE_ID_AUTO;
UCBioAPI_RETURN err = UCBioAPI_CloseDevice(m_hUCBioAPI, nDeviceID);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to close device
} else {
   // Failed to close device
}
```

3.4.4. 디바이스 정보 얻기

현재 Open한 디바이스에 대한 정보를 얻고자 할 경우 UCBioAPI_GetDeviceInfo 함수를 사용 할 수 있다. 디바이스가 지원하는 이미지의 크기 등에 대한 정보를 얻을 수 있다. (중요하지 않은 코드부분은 회색으로 처리됨)

■ **Example**

```
UCBioAPI_UINT32    m_nDeviceWidth, m_nDeviceHeight;
…
int nDeviceID = UCBioAPI_DEVICE_ID_AUTO;
UCBioAPI_RETURN err = UCBioAPI_OpenDevice(m_hUCBioAPI, nDeviceID);
if (err == UCBioAPIERROR_NONE) {
   UCBioAPI_DEVICE_INFO_0 deviceInfo0;
   memset(&deviceInfo0, 0, sizeof(UCBioAPI_DEVICE_INFO_0));
   err = UCBioAPI_GetDeviceInfo(m_hUCBioAPI, nDeviceID, 0,
                                &deviceInfo0);
   if (err == UCBioAPIERROR_NONE) {
      m_nDeviceWidth = deviceInfo0.ImageWidth;
      m_nDeviceHeight = deviceInfo0.ImageHeight;
   }
} else {
   // Failed to open device
}
```

3.5. FIR 데이터의 이해

지문을 등록하고 인증하기 위해서는 반드시 UCBioBSP SDK가 사용하는 지문 데이터인 FIR에 대한 이해가 필요하다. 1장에서 지문 데이터의 구조에 대해서는 설명을 했지만 실제 FIR을 사용하기 위해서 알아야 할 것들에 대해 설명한다.

3.5.1. FIR의 종류

FIR은 다음의 세가지 종류로 구분 될 수 있다.

■ **FIR Handle**

UCBioBSP SDK에서 제공하는 API를 이용해 지문을 등록하거나 이미지를 획득할 경우 사용자가 얻게 되는 것은 실제 FIR 데이터가 아니라 FIR Handle값을 얻게 된다. 이 FIR Handle은 BSP 내부적으로 메모리가 관리되므로 사용자가 등록한 지문 데이터를 DB에 저장 하거나 네트워크를 통해 전송하고자 할 경우 반드시 실제 FIR 데이터를 Handle로부터 얻어와야만 한다. 실제 FIR 데이터를 얻기 위해서는 UCBioAPI_GetFIRFromHandle 등과 같은 함수를 사용해야 한다. Handle도 사용이 끝난 경우 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리 해제를 해 주어야 한다.

■ **FIR (Full FIR)**

FIR Handle로부터 얻어온 실제 지문 데이터에 대한 암호화된 바이너리 메모리 블록을 포함한 구조체이다. 이 값을 얻기 위해서는 UCBioAPI_GetFIRFromHandle과 같은 함수를 사용 할 수 있다. 이 구조체에 대한 설명은 1장에 있으므로 참조하면 된다.

■ **Text encoded FIR**

FIR Handle로부터 얻어온 실제 지문 데이터에 대한 암호화된 문자열 형태의 메모리 블록을 포함한 구조체이다. 이 값을 얻기 위해서는 UCBioAPI_GetTextFIRFromHandle과 같은 함수를 사용 할 수 있다. 유니코드를 위해 ANSI 문자열과 유니코드를 모두 지원 가능하다.

3.5.2. FIR의 사용

인증 등의 함수에서 FIR을 사용하기 위해서는 UCBioAPI_INPUT_FIR이라는 구조체를 이용해야 한다. 이 구조체는 내부적으로 union 구조체를 포함하고 있어 위에서 설명한 세가지 종류의 FIR을 모두 포함 할 수 있는 구조로 되어 있다.

```
typedef struct ucbioapi_input_fir {
  UCBioAPI_INPUT_FIR_FORM        Form;
  union {
    UCBioAPI_FIR_HANDLE_PTR      FIRinBSP;
    UCBioAPI_VOID_PTR            FIR;
    UCBioAPI_FIR_TEXTENCODE_PTR  TextFIR;
  } InputFIR;
} UCBioAPI_INPUT_FIR, *UCBioAPI_INPUT_FIR_PTR;
```

FIR 종류에 따른 각각의 사용법은 다음과 같다. (중요하지 않은 코드부분은 회색으로 처리됨)

■ **FIR Handle**

```
UCBioAPI_FIR_HANDLE hFIR;
UCBioAPI_RETURN err = UCBioAPI_Capture(m_hUCBioAPI,
                                       UCBioAPI_FIR_PURPOSE_VERIFY,
                                       &hFIR,
                                       UCBioAPI_USE_DEFAULT_TIMEOUT,
                                       NULL,
                                       NULL);
…
UCBioAPI_INPUT_FIR inputFIR;
inputFIR.Form = UCBioAPI_FIR_FORM_HANDLE;
inputFIR.InputFIR.FIRinBSP = &hFIR;
…
UCBioAPI_BOOL bResult;
err = UCBioAPI_Verify(m_hUCBioAPI, &inputFIR, &bResult, NULL,
                      UCBioAPI_USE_DEFAULT_TIMEOUT, NULL, NULL);
…
UCBioAPI_FreeFIRHandle(hFIR);
```

■ **FIR (Full FIR)**

```
UCBioAPI_FIR fullFIR;
UCBioAPI_GetFIRFromHandle(hFIR, &fullFIR);
…
UCBioAPI_INPUT_FIR inputFIR;
inputFIR.Form = UCBioAPI_FIR_FORM_FULLFIR;
inputFIR.InputFIR.FIR = &fullFIR;
…
UCBioAPI_BOOL bResult;
err = UCBioAPI_Verify(m_hUCBioAPI, &inputFIR, &bResult, NULL,
                      UCBioAPI_USE_DEFAULT_TIMEOUT, NULL, NULL);
…
UCBioAPI_FreeFIR(&fullFIR);
```

■ **Text encoded FIR**

```
UCBioAPI_FIR_TEXTENCODE textFIR;
UCBioAPI_GetTextFIRFromHandle(hFIR, &textFIR, UCBioAPI_FALSE);
…
UCBioAPI_INPUT_FIR inputFIR;
```

```
inputFIR.Form = UCBioAPI_FIR_FORM_TEXTENCODE;
inputFIR.InputFIR.TextFIR = &textFIR;
…
UCBioAPI_BOOL bResult;
err = UCBioAPI_Verify(m_hUCBioAPI, &inputFIR, &bResult, NULL,
                      UCBioAPI_USE_DEFAULT_TIMEOUT, NULL, NULL);
…
UCBioAPI_FreeTextFIR(&textFIR);
```

3.5.3. FIR 메모리 해제

FIR의 사용이 끝나고 더 이상 사용할 필요가 없을 경우에는 메모리를 해제해 주어야 한다.
FIR 종류에 따른 각각의 메모리 해제 방식은 다음과 같다.

■ **FIR Handle**
UCBioAPI_FreeFIRHandle 함수를 이용해 메모리를 해제해야 한다.

■ **FIR (Full FIR)**
UCBioAPI_FreeFIR 함수를 이용해 메모리를 해제해야 한다.

■ **Text encoded FIR**
UCBioAPI_FreeTextFIR 함수를 이용해 메모리를 해제해야 한다.

각 해제방식에 대한 예제 코드는 위의 FIR 사용에서 설명된 예제를 참조하기 바란다.

3.5.4. FIR의 변환

FIR을 저장하거나 네트워크를 통해 전송하기 위해서는 Handle로는 안되고 반드시 Full FIR이나 Text encoded FIR로 바뀌어야 한다. 두 경우 모두 구조체 내부에 포인터를 포함하고 있으므로 저장이나 전송을 위해서는 Stream 형태의 데이터로 적절히 변환하는 것이 필요하다.

다음은 FIR 종류에 따른 각각의 데이터를 Stream 형태의 데이터로 변환하는 예이다.

■ **FIR (Full FIR)**

```
UCBioAPI_FIR fullFIR;
UCBioAPI_GetFIRFromHandle(hFIR, &fullFIR);
…
UINT nHeaderLength = sizeof(fullFIR.Format) + fullFIR.Header.Length;
UINT nStreamLength = nHeaderLength + fullFIR.Header.DataLength;
BYTE* pFIRStream = new BYTE [nStreamLength];
If (pFIRStream) {
```

```
   memset(pFIRStream, 0, nStreamLength);
   memcpy(pFIRStream, &fullFIR, nHeaderLength);
   memcpy(pFIRStream+nHeaderLength, fullFIR.Data,
          fullFIR.Header.DataLength);
}
…
if (pFIRStream)
   delete[] pFIRStream;
```

■ **Text encoded FIR**

```
UCBioAPI_FIR_TEXTENCODE textFIR;
UCBioAPI_GetTextFIRFromHandle(hFIR, &textFIR, UCBioAPI_FALSE);
…
UINT nHeaderLength = sizeof(textFIR.IsWideChar);
UINT nStringLength = strlen(textFIR.TextFIR);
UINT nStreamLength = nHeaderLength + nStringLength + 1;
BYTE* pFIRStream = new BYTE [nStreamLength];
If (pFIRStream) {
   memset(pFIRStream, 0, nStreamLength);
   memcpy(pFIRStream, &textFIR, nHeaderLength);
   memcpy(pFIRStream+nHeaderLength, textFIR.TextFIR, nStringLength);
}
…
if (pFIRStream)
   delete[] pFIRStream;
```

3.6. 지문 등록하기

지문을 등록하기 위해서는 반드시 먼저 디바이스가 Open되어 있어야 한다.
UCBioBSP SDK에서는 등록된 지문이 FIR Handle의 형태로 얻어지게 된다. FIR Handle은 BSP 내부에서 메모리가 관리되므로 만약 FIR Handle로부터 실제 지문 데이터를 얻고자 한다면 FIR 관리 함수들을 사용하여야 한다.

3.6.1. 신규 지문 등록 및 기존 지문 수정

지문을 등록하기 위해서는 UCBioAPI_Enroll 함수를 사용한다. 지문 등록 함수는 등록뿐만 아니라 이미 기존에 등록된 지문을 수정 할 수 있는 기능도 가지고 있다. 지문 수정을 위해서는 두 번째 인자에 기존 지문을 지정하면 된다. 새로운 지문을 등록할 경우에는 이 값을 NULL로 지정하면 된다.

3.6.2. Payload 지정

지문 등록 시 Payload를 지정해 등록된 지문 데이터 내부에 사용자의 특정 데이터를 암호화해서 삽입 할 수도 있다. 이렇게 등록된 Payload는 지문 인증이 성공 할 경우에만 얻어 올 수 있다.

3.6.3. 자동 지문 감지(Auto Detect)

지문 입력장치의 기본 기능은 지문을 입력받을 때 LED를 켬으로서 지문입력을 유도하게 된다. 그러나 이 기능을 사용하지 않고 지문을 입력장치에 가져다 댄 경우 자동으로 지문을 획득하기 위한 방식으로 모든 지문입력 명령 전에 지시하면 된다.

3.6.4. 사용 예

아래 예제에서는 기존 등록 데이터를 수정하면서 새로운 Payload 데이터로 바꾸는 것을 알 수 있다. 좀 더 자세한 예제는 SDK 설치 후 설치 폴더내의 Samples 폴더내의 UCBioBSP_Demo 폴더에서 찾아 볼 수 있다.

■ **Example**

```
UCBioAPI_FIR_HANDLE       m_hEnrolledFIR
…

UCBioAPI_FIR_HANDLE hNewFIR;

UCBioAPI_INPUT_FIR inputFIR;
inputFIR.Form = UCBioAPI_FIR_FORM_HANDLE;
inputFIR.InputFIR.FIRinBSP = &m_hEnrolledFIR;  // Existing FIR

UCBioAPI_FIR_PAYLOAD payload;
payload.Length = (lstrlen(m_szPayload) + 1) * sizeof(TCHAR);
payload.Data = new UCBioAPI_UINT8 [payload.Length];
memset(payload.Data, 0, payload.Length);
```

```
lstrcpy((LPTSTR)payload.Data, m_szPayload);

UCBioAPI_RETURN err = UCBioAPI_Enroll(m_hUCBioAPI, &inputFIR, &hNewFIR,
                    &payload, UCBioAPI_USE_DEFAULT_TIMEOUT, NULL, NULL);

if (err == UCBioAPIERROR_NONE) {
   UCBioAPI_FreeFIRHandle(m_hEnrolledFIR);
   m_hEnrolledFIR = hNewFIR;      // Replace new FIR to existing FIR
}
if (payload.Data)
   delete[] payload.Data;

// Auto Detect 기능 설정(FALSE:기존방식, TRUE:AutoDetect)
UCBioAPI_RETURN err = UCBioAPI_SetAutoDetect(m_hUCBioAPI, TRUE);
```

3.7. 지문 획득하기

지문을 획득하기 위해서는 반드시 먼저 디바이스가 Open되어 있어야 한다.
지문을 획득하는 것은 지문 등록과 마찬가지로 획득한 지문 데이터를 FIR Handle로 얻게 되지만 등록과는 달리 단지 하나의 라이브 지문만을 입력 받는다. (단, 획득 목적을 등록용으로 하게 되면 지문 등록과정과 동일한 과정으로 진행된다.)
이렇게 획득된 지문 데이터와 이전에 등록 받은 지문 데이터를 비교하여 지문 인증을 수행 할 수 있다.

3.7.1. 지문 획득

지문을 획득하기 위해서는 UCBioAPI_Capture 함수를 사용한다. 획득을 위한 목적을 지정 할 수 있는데 사용 할 수 있는 값으로는 다음과 같다.

```
#define UCBioAPI_FIR_PURPOSE_VERIFY                              (0x01)
#define UCBioAPI_FIR_PURPOSE_IDENTIFY                            (0x02)
#define UCBioAPI_FIR_PURPOSE_ENROLL                              (0x03)
#define UCBioAPI_FIR_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY        (0x04)
#define UCBioAPI_FIR_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY      (0x05)
#define UCBioAPI_FIR_PURPOSE_AUDIT                               (0x06)
#define UCBioAPI_FIR_PURPOSE_UPDATE                              (0x10)
```

만약 등록용 목적을 지정 할 경우 UCBioAPI_Enroll 함수를 부르는 것과 동일한 효과를 가진다.
지정된 목적은 참고용으로만 사용되며 향후 인증에 영향을 미치지는 않는다.

3.7.2. 사용 예

■ **Example**

```
UCBioAPI_FIR_HANDLE hCapturedFIR
UCBioAPI_RETURN err = UCBioAPI_Capture(m_hUCBioAPI,
                                       UCBioAPI_FIR_PURPOSE_VERIFY,
                                       &hCapturedFIR,
                                       UCBioAPI_USE_DEFAULT_TIMEOUT,
                                       NULL, NULL);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to capture FIR
} else {
   // Failed to capture FIR
}
```

3.8. 지문 인증하기

지문을 인증하기 위해서는 반드시 디바이스가 있어야 하는 것은 아니지만 라이브 지문과의 인증을 위해서는 반드시 먼저 디바이스가 Open 되어 있어야 한다.
지문 인증을 위해서는 크게 아래 두 가지 방식이 사용된다.

3.8.1. 라이브 지문과 등록 지문과의 인증

이미 등록 받은 지문 데이터를 입력 값으로 주고 현재 지문인식기로부터 지문을 라이브로 입력 받아 서로 비교하는 방식이다. 때문에 반드시 디바이스가 Open된 상태로 사용하여야 한다.
사용하는 함수로는 UCBioAPI_Verify 함수를 사용한다.

■ **Example**

```
UCBioAPI_FIR_HANDLE hEnrolledFIR;
UCBioAPI_RETURN err = UCBioAPI_Enroll(m_hUCBioAPI,
                                      NULL,
                                      &hEnrolledFIR,
                                      NULL,
                                      UCBioAPI_USE_DEFAULT_TIMEOUT,
                                      NULL,
                                      NULL);
…
UCBioAPI_INPUT_FIR inputFIR;
inputFIR.Form = UCBioAPI_FIR_FORM_HANDLE;
inputFIR.InputFIR.FIRinBSP = &hEnrolledFIR;
…
UCBioAPI_BOOL bResult;
err = UCBioAPI_Verify(m_hUCBioAPI, &inputFIR, &bResult, NULL,
                      UCBioAPI_USE_DEFAULT_TIMEOUT, NULL, NULL);

if (err == UCBioAPIERROR_NONE && bResult) {
   // Succeeded to verify
} else {
   // Failed to verify
}
…
```

3.8.2. 미리 획득한 지문과 등록 지문과의 인증

이미 등록 받은 지문 데이터와 미리 획득한 지문 데이터를 입력 값으로 주고 그 두 데이터를 비교하는 방식이다. 때문에 디바이스가 없어도 사용이 가능하다. 서버에서 클라이언트로부터 전송되어져 온 지문 데이터의 인증만을 할 경우 사용하는 방식이다.

사용하는 함수로는 UCBioAPI_VerifyMatch 함수를 사용한다.

■ **Example**

```
UCBioAPI_FIR_HANDLE hEnrolledFIR;
UCBioAPI_RETURN err = UCBioAPI_Enroll(m_hUCBioAPI,
                                      NULL,
                                      &hEnrolledFIR,
                                      NULL,
                                      UCBioAPI_USE_DEFAULT_TIMEOUT,
                                      NULL,
                                      NULL);
…
UCBioAPI_FIR_HANDLE hCapturedFIR
UCBioAPI_RETURN err = UCBioAPI_Capture(m_hUCBioAPI,
                                      UCBioAPI_FIR_PURPOSE_VERIFY,
                                      &hCapturedFIR,
                                      UCBioAPI_USE_DEFAULT_TIMEOUT,
                                      NULL, NULL);
…
UCBioAPI_INPUT_FIR inputEnrolledFIR;
inputEnrolledFIR.Form = UCBioAPI_FIR_FORM_HANDLE;
inputEnrolledFIR.InputFIR.FIRinBSP = &hEnrolledFIR;

UCBioAPI_INPUT_FIR inputCapturedFIR;
inputCapturedFIR.Form = UCBioAPI_FIR_FORM_HANDLE;
inputCapturedFIR.InputFIR.FIRinBSP = &hCapturedFIR;
…
UCBioAPI_BOOL bResult;
err = UCBioAPI_VerifyMatch(m_hUCBioAPI,
                           &inputEnrolledFIR, &inputCapturedFIR,
                           &bResult, NULL);

if (err == UCBioAPIERROR_NONE && bResult) {
   // Succeeded to verify
} else {
   // Failed to verify
}
…
```

3.8.3. Payload 데이터 얻기

지문을 인증이 성공하면 지문 등록 시 삽입했던 Payload를 얻을 수 있다. Payload 데이터는 어떠한 데이터도 사용 될 수 있기 때문에 사용자 인증 시 특정 고정된 값을 얻고자 할 경우 사용 할 수 있다. 아래 예는 인증 시 Payload를 얻어 오도록 한다.

■ **Example**

```
…
UCBioAPI_BOOL bResult;
UCBioAPI_FIR_PAYLOAD payload;
err = UCBioAPI_VerifyMatch(m_hUCBioAPI,
                           &inputEnrolledFIR, &inputCapturedFIR,
                           &bResult, &payload);

if (err == UCBioAPIERROR_NONE && bResult) {
   // Succeeded to verify
   // Use payload data…
   …
   // Free payload data
   UCBioAPI_FreePayload(&payload);
} else {
   // Failed to verify
}
…
```

3.9. FastSearch (1:N 인증) 사용하기

UCBioBSP SDK는 1:N의 고속 인증을 위해 FastSearch Engine을 제공한다. 많은 수의 사용자를 인증할 경우 단순히 1:1 인증의 반복을 통해서는 효과적인 인증 속도를 기대하기 어렵다. 때문에 1:N만을 위한 인증이 필요하게 되고 FastSearch Engine을 통해 다수의 사용자 중 1명을 인증하는 기능을 제공하게 되었다. 이 장에서는 FastSearch용 API에 대한 설명을 한다. FastSearch를 사용하기 위해서는 UCBioAPI_FastSearch.h 파일을 include해서 사용해야 한다.

3.9.1. 초기화 및 종료

FastSearch Engine을 초기화 하기 위해서는 UCBioAPI_FastSearchEngine 함수를 호출해야 한다. 이 함수를 호출하면 이후부터 FastSearch Engine을 사용 할 수 있는 상태가 된다.
모든 작업이 완료되면 FastSearch Engine을 종료해 주어야 하는데 그때는 UCBioAPI_TerminateFastSearchEngine 함수를 호출해 주면 된다.

■ **Example**

```
#include "UCBioAPI_FastSearch.h"
…
// Initialize FastSearch Engine
err = UCBioAPI_InitFastSearchEngine(m_hUCBioAPI);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to initialize
   …

   // Terminate FastSearch Engine
   UCBioAPI_TerminateFastSearchEngine(m_hUCBioAPI);
} else {
   // Failed to verify
}
```

3.9.2. 기본 설정 값 얻어오기 및 새로운 값 설정

FastSearch Engine의 기본 설정 값을 얻어오거나 새로운 값을 설정 할 수 있다.
기본 설정 값을 얻어오기 위해서는 UCBioAPI_GetFastSearchInitInfo 함수를 사용하고 값을 바꾸기 위해서는 UCBioAPI_SetFastSearchInitInfo 함수를 사용하면 된다. 각 설정값에 대한 자세한 사항은 함수 레퍼런스를 참조하면 된다.

3.9.3. DB 만들기

1:N 인증을 수행하기 위해서는 우선 인증을 수행할 다수의 DB를 메모리상에 먼저 만들어야 한다. 고속의 인증을 위해 내부적으로 메모리 DB를 만들어 인증을 수행하게 된다. 그러기 위해서

는 각각의 FIR 데이터를 하나의 인증용 메모리 DB로 묶는 작업이 필요한데 UCBioAPI_AddFIRToFastSearchDB 함수는 입력 받은 FIR 데이터나 사용자 DB에 등록되어 있는 FIR 데이터를 하나의 메모리 DB로 만들어 준다.
또한 FastSearch Engine은 내부적으로 FIR을 하나의 데이터 단위로 사용하지 않고 Template 단위를 데이터 단위로 사용한다. 때문에 만약 FIR 내부에 여러 개의 Template이 포함된 경우라면 하나의 FIR을 DB에 추가하더라도 내부적으로 여러 개의 Template이 DB에 추가되어질 수 있다.
입력값으로는 DB에 등록할 FIR 데이터와 그 데이터에 대한 사용자 ID 값을 넘겨준다. 향후 1:N 인증이 성공하면 이때 넘겨준 사용자 ID 값을 얻을 수 있다.

■ **Example**

```
…
err = UCBioAPI_AddFIRToFastSearchDB(m_hUCBioAPI, &inputEnrolledFIR,
                                    nUserID, NULL);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to add FIR
   …
} else {
   // Failed to add FIR
}
…
```

3.9.4. 메모리 DB 관리

만들어진 인증용 메모리 DB를 관리하기 위해 다음과 같은 다양한 함수가 제공된다.

■ **UCBioAPI_RemoveFpFromFastSearchDB**
메모리 DB에서 특정 지문 하나를 삭제한다.

■ **UCBioAPI_RemoveUserFromFastSearchDB**
메모리 DB에서 특정 사용자의 지문을 모두 삭제한다. 메모리 DB에는 특정 사용자에 대해 다수의 지문 정보가 존재 할 수 있으므로 이것을 한꺼번에 삭제 하고자 할 경우 유용하다.

■ **UCBioAPI_ClearFastSearchDB**
메모리 DB 전체를 지운다.

■ **UCBioAPI_GetFpCountFromFastSearchDB**
메모리 DB에 있는 지문의 개수를 얻어온다.

■ **UCBioAPI_GetFpInfoFromFastSearchDB**
메모리 DB에 있는 특정 위치의 지문 정보를 얻어온다.

■ **UCBioAPI_CheckFpExistInFastSearchDB**
특정 지문 정보가 메모리 DB에 있는지를 검사한다.

■ **UCBioAPI_SaveFastSearchDBToFile**
메모리 DB 전체를 파일로 저장한다. 이렇게 저장해둔 DB는 LoadFastSearchDBFromFile 함수를 이용해 다시 읽을 수 있다.

■ **UCBioAPI_LoadFastSearchDBFromFile**
File로 저장된 DB를 메모리 DB로 다시 읽어 들인다.
이렇게 할 경우 UCBioAPI_AddFIRToFastSearchDB 함수를 이용해 새로 DB를 구축하는 것 보다 훨씬 빠르게 메모리 DB를 만들 수가 있다.

이 함수들에 대한 좀 더 자세한 설명은 함수 레퍼런스를 참고하기 바라며 자세한 사용 예는 SDK 설치 후 Samples 폴더내의 UCBioBSP_FastSearchDemo 폴더에서 찾을 수 있다.

3.9.5. 1:N 인증하기

만들어진 메모리 지문 DB에서 특정 사용자의 지문을 인증하기 위해서는 UCBioAPI_IdentifyFIRFromFastSearchDB 함수를 사용하면 된다.
인증 시에는 인증을 위한 보안 레벨값을 정할 수 있으며 매 인증 시마다 정보를 얻을 수 있는 Callback 함수도 등록해 사용 할 수 있다.

■ **Example**

Callback Function 부분

```
UCBioAPI_RETURN WINAPI MyFastSearchCallBack
             (UCBioAPI_FASTSEARCH_CALLBACK_PARAM_PTR_0 pCallbackParam0,
              UCBioAPI_VOID_PTR pUserParam)
{
   MyInfo* pInfo = (MyInfo*)pUserParam;
   …

   if (pInfo->m_bStopFlag)
      return UCBioAPI_FASTSEARCH_CALLBACK_STOP;
   else
      return UCBioAPI_FASTSEARCH_CALLBACK_OK;
}
```

Identify 부분

```
…

UCBioAPI_FIR_HANDLE hFIR;
```

```
ret = UCBioAPI_Capture(m_hUCBioBSP, UCBioAPI_FIR_PURPOSE_IDENTIFY,
                       &hFIR, UCBioAPI_USE_DEFAULT_TIMEOUT, NULL, NULL);
if (ret == UCBioAPIERROR_NONE) {
   UCBioAPI_FASTSEARCH_FP_INFO infoFp;

   UCBioAPI_INPUT_FIR inputFIR;
   inputFIR.Form = UCBioAPI_FIR_FORM_HANDLE;
   inputFIR.InputFIR.FIRinBSP = &hFIR;

   UCBioAPI_FASTSEARCH_CALLBACK_INFO_0 callbackInfo0;
   callbackInfo0.CallBackType = 0;
   callbackInfo0.CallBackFunction = MyFastSearchCallBack;
   callbackInfo0.UserCallBackParam = &myInfo;

   ret = UCBioAPI_IdentifyFIRFromFastSearchDB(pDlg->m_hUCBioBSP,
                                  &inputFIR, 5, &infoFp, &callbackInfo0);

   if (ret != UCBioAPIERROR_NONE) {
      if (ret == UCBioAPIERROR_FASTSEARCH_IDENTIFY_STOP) {
         // User stop!
      } else {
         // Failed to identify fingerprint data from DB!
      }
   } else {
      // Succeeded to identify
   }
   UCBioAPI_FreeFIRHandle(hFIR);
}
…
```

위 예제에서는 Callback 함수를 등록해 Identify를 수행 중 매 인증이 일어날 때 마다 현재 인증하고 있는 지문 정보와 함께 Callback 함수가 불리게 된다. 사용자는 이 Callback 함수를 통해 현재 진행 상태를 표시 할 수도 있으며 중도에 인증을 중단 시킬 수도 있다.
인증이 성공하면 인자로 넘겨받은 infoFp 구조체를 통해 인증된 지문 정보를 얻을 수 있게 된다. 자세한 사용 예는 SDK 설치 후 Samples 폴더내의 UCBioBSP_FastSearchDemo 폴더에서 찾을 수 있다.

3.10. FIR 데이터 변환하기

앞에서도 언급했듯이 FIR 데이터는 다수의 Template 데이터로 구성된 데이터의 집합이다. 때문에 FIR 데이터로부터 각각의 Template 데이터를 독립적으로 얻고자 할 경우나 또는 다수의 Template 데이터를 이용해 하나의 FIR 데이터를 만들고자 할 경우에는 변환 함수를 사용해야 한다.
더불어 이미지를 담는 Audit FIR의 경우에도 제공되는 변환 함수들을 이용하여 각각의 손가락에 대한 Raw 이미지를 얻을 수 있다.
이 장에 소개된 함수들을 사용하기 위해서는 UCBioAPI_Export.h 파일을 include하여 사용하여야 한다.

3.10.1. FIR 데이터로부터 Template 데이터 추출

FIR 데이터로부터 Template 데이터를 추출하기 위해서는 UCBioAPI_FIRToTemplate 함수를 사용한다. 이 함수를 사용하면 Template 데이터뿐만 아니라 FIR의 각종 정보들도 얻을 수 있다. 이렇게 얻어진 UCBioAPI_EXPORT_DATA는 사용이 끝난 후 반드시 UCBioAPI_FreeExportData 함수를 이용해 메모리 해제를 해 주어야 한다.

■ **Example**

```
#include "UCBioAPI_Export.h"
…
UCBioAPI_EXPORT_DATA exportData;
err = UCBioAPI_FIRToTemplate(m_hUCBioAPI, &inputFIR, &exportData,
                             UCBioAPI_TEMPLATE_TYPE_SIZE400);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to export data
   …

   // Free export data
   UCBioAPI_FreeExportData(&exportData);
} else {
   // Failed to export data
}
```

3.10.2. Template 데이터를 이용해 FIR Handle 만들기

Template 데이터로부터 FIR Handle을 만들기 위해서는 다음의 3가지 함수가 사용될 수 있다.

■ **UCBioAPI_TemplateToFIR**

일반적으로 Template 데이터는 지문 특징점 정보만 가질 뿐 FIR과 달리 손가락 정보라든지 그 외 다른 정보는 가지지 않는다. 때문에 단순히 Template 데이터를 FIR Handle로만 단순 변환 하여 인증을 위해 사용 할 경우에는 이 함수를 사용하면 된다. 이 경우 FIR 내부적으로

는 손가락 정보나 기타 관련 정보는 들어가지 않게 된다.

■ **UCBioAPI_TemplateToFIREx**

UCBioAPI_TemplateToFIR 함수와 거의 동일하나 여러 개의 Template을 배열로 지정해 하나의 FIR로 만들 수 있다. 이때 각 Template의 크기는 모두 동일해야 하며 그 길이값은 함수의 인자로 넘겨주어 알려주게 된다.

```
UCBioAPI_UINT8 pMyTemplateData[400*2];
// Load template data to pMyTemplateData
…

UCBioAPI_FIR_HANDLE hNewFIR;
err = UCBioAPI_TemplateToFIREx(m_hUCBioAPI, pMyTemplateData,
                               400*2, 400,
                               UCBioAPI_TEMPLATE_TYPE_SIZE400,
                               UCBioAPI_FIR_PURPOSE_VERIFY, &hNewFIR);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to import data
   …
} else {
   // Failed to import data
}
```

■ **UCBioAPI_ImportDataToFIR**

UCBioAPI_EXPORT_DATA 구조체를 이용해 지문 정보를 모두 구축한 후 FIR을 만들 경우 사용하는 함수이다. 이 경우 정확한 데이터 값을 모두 지정가능 하기 때문에 원하는 FIR Handle을 정확히 만들어 낼 수 있다.

```
…
int fn = 1, sn = 1;

UCBioAPI_EXPORT_DATA exportData;
memset(&exportData, 0, sizeof(UCBioAPI_EXPORT_DATA));

exportData.Length = sizeof(UCBioAPI_EXPORT_DATA);
exportData.TemplateType = UCBioAPI_TEMPLATE_TYPE_SIZE400;
exportData.FingerNum = fn;
exportData.DefaultFingerID = UCBioAPI_FINGER_ID_RIGHT_THUMB;
exportData.SamplesPerFinger = sn;
exportData.FingerInfo = new UCBioAPI_FINGER_BLOCK [fn];
exportData.FingerInfo[0].Length = sizeof(UCBioAPI_FINGER_BLOCK);
```

```
exportData.FingerInfo[0].FingerID = UCBioAPI_FINGER_ID_RIGHT_THUMB;
exportData.FingerInfo[0].TemplateInfo =new UCBioAPI_TEMPLATE_BLOCK[sn];
exportData.FingerInfo[0].TemplateInfo[0].Length = 400;
exportData.FingerInfo[0].TemplateInfo[0].Data =new UCBioAPI_UINT8[400];
memcpy(exportData.FingerInfo[0].TemplateInfo[0].Data,
                                                pMyTemplateData, 400);

UCBioAPI_FIR_HANDLE hNewFIR;
err = UCBioAPI_ImportDataToFIR(m_hUCBioAPI, &exportData,
                               UCBioAPI_FIR_PURPOSE_VERIFY, &hNewFIR);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to import data
   …

   // Free export data
   if (exportData.FingerInfo[0].TemplateInfo)
      delete[]exportData.FingerInfo[0].TemplateInfo;
   if (exportData.FingerInfo)
      delete exportData.FingerInfo;
} else {
   // Failed to import data
}
```

3.10.3. Template 데이터 간의 변환

다양한 형식의 Template 데이터 간의 변환은 UCBioAPI_ConvertTemplateData 함수를 사용한다. 이렇게 변환된 데이터는 사용 후 반드시 UCBioAPI_FreeData 함수를 이용해 메모리 해제를 해 주어야 한다.

상호 변환 할 수 있는 Template의 형태로는 현재 아래와 같은 값이 사용될 수 있다.

```
#define UCBioAPI_TEMPLATE_TYPE_SIZE400          (400)
#define UCBioAPI_TEMPLATE_TYPE_SIZE800          (800)
#define UCBioAPI_TEMPLATE_TYPE_SIZE320          (320)
#define UCBioAPI_TEMPLATE_TYPE_SIZE256          (256)
#define UCBioAPI_TEMPLATE_TYPE_FMR              (1)
```

3.10.4. Audit FIR 데이터로부터 Raw 이미지 추출

Audit FIR은 UCBioAPI_Capture나 UCBioAPI_Enroll 함수 사용 시 Audit Data라고 해서 얻어지는 이미지 정보를 담고 있는 FIR 데이터이다. 이 Audit FIR은 일반 FIR과 동일한 구조를 가지지만 내부적으로 이미지를 담고 있다는 것만 다르다. 이 Audit FIR 데이터로부터 Raw 이미지를 추출하

기 위해서는 UCBioAPI_AuditFIRToImage 함수를 사용하면 된다.
이렇게 얻어진 UCBioAPI_EXPORT_AUDIT_DATA는 사용이 끝난 후 UCBioAPI_FreeExportAuditData 함수를 이용해 메모리 해제를 해 주어야 한다.

■ **Example**

```
#include "UCBioAPI_Export.h"
…
UCBioAPI_EXPORT_AUDIT_DATA exportAuditData;
err = UCBioAPI_AuditFIRToImage(m_hUCBioAPI, &inputFIR,
                               &exportAuditData);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to export audit data
   …

   // Free export audit data
   UCBioAPI_FreeExportAuditData(&exportAuditData);
} else {
   // Failed to export audit data
}
```

3.10.5. Raw 이미지를 이용해 Audit FIR Handle 만들기

Raw 이미지를 이용해 Audit FIR Handle을 만들기 위해서는 UCBioAPI_ImageToAuditFIR 함수를 사용하면 된다.

3.11. UI 설정하기

UCBioBSP SDK에서 사용되는 사용자 인터페이스를 구성하기 위한 방법에 대해 설명한다.

3.11.1. Skin 파일 불러오기

UCBioBSP SDK에서는 사용하는 등록 및 인증용 화면을 Skin 방식의 UI로 사용하고 있다. 때문에 UCBioBSP SDK에서 기본 제공하는 UI를 사용하지 않고 다른 언어로 된 UI나 또는 다른 형태의 UI를 사용하고자 하는 경우에는 사용자 정의 Skin을 만들어 사용 할 수 있다. 이때 만들어진 Skin DLL을 읽어 사용하는 함수가 UCBioAPI_SetSkinResource 함수이다.
현재 UCBioBSP SDK는 기본적으로 영문으로 된 Skin을 내장하고 있다. 만약 이것을 한글로 된 Skin으로 바꾸고자 할 경우에는 다음과 같이 하면 된다.

■ **Example**

```
err = UCBioAPI_SetSkinResource("UCBioBSPSkin_Kor.dll");
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to change to new skin
} else {
   // Failed to change to new skin
}
```

사용자 정의 Skin을 만들기 위해서는 본사로 문의하면 된다.

3.11.2. UI 속성 변경하기

UCBioBSP SDK에서는 사용자가 임의로 UI 관련 속성을 변경해 작업 할 수 있는 기능을 제공하고 있다. 이 기능은 UCBioAPI_WINDOW_OPTION이라는 구조체를 UCBioAPI_Enroll 함수나 UCBioAPI_Capture, UCBioAPI_Verify 함수에 인자로 넘겨 사용 할 수 있다.

■ **UCBioAPI_WINDOW_OPTION**

```
typedef struct ucbioapi_window_option {
   UCBioAPI_UINT32                Length;
   UCBioAPI_WINDOW_STYLE          WindowStyle;
   UCBioAPI_HWND                  ParentWnd;
   UCBioAPI_HWND                  FingerWnd;
   UCBioAPI_CALLBACK_INFO_0       CaptureCallBackInfo;
   UCBioAPI_CALLBACK_INFO_1       FinishCallBackInfo;
   UCBioAPI_CHAR_PTR              CaptionMsg;
   UCBioAPI_CHAR_PTR              CancelMsg;
   UCBioAPI_WINDOW_OPTION_PTR_2   Option2;
} UCBioAPI_WINDOW_OPTION, *UCBioAPI_WINDOW_OPTION_PTR;
```

```
typedef struct ucbioapi_window_option_2 {
  UCBioAPI_UINT8                FPForeColor[3];
  UCBioAPI_UINT8                FPBackColor[3];
  UCBioAPI_UINT8                DisableFingerForEnroll[10];
  UCBioAPI_UINT32               Reserved1[4];
  UCBioAPI_VOID_PTR             Reserved2;
} UCBioAPI_WINDOW_OPTION_2, *UCBioAPI_WINDOW_OPTION_PTR_2;
```

각 구조체 멤버에 대한 설명은 다음과 같다.

■ **Length**
구조체의 길이값. 현재는 sizeof(UCBioAPI_WINDOW_OPTION) 값과 같다.

■ **WindowStyle**
윈도우가 화면에 표시되는 형태를 지정 할 수 있다. 윈도우가 팝업 형태로 뜰 것인지 아니면 다른 윈도우의 영역에 지문만 표시 할 것인지를 지정 할 수 있고 기타 Flag값들도 추가적으로 지정해 사용 할 수 있다. 가질 수 있는 값들은 다음과 같다.

```
#define UCBioAPI_WINDOW_STYLE_POPUP           (0)
#define UCBioAPI_WINDOW_STYLE_INVISIBLE       (1)

#define UCBioAPI_WINDOW_STYLE_NO_FPIMG        (0x00010000)
#define UCBioAPI_WINDOW_STYLE_NO_WELCOME      (0x00020000)
#define UCBioAPI_WINDOW_STYLE_NO_TOPMOST      (0x00040000)
```

아래 세가지 Flag는 중복해서 사용 가능하므로 OR 연산자를 이용해 지정 가능하다.
각 Flag에 대한 자세한 설명은 API 레퍼런스를 참조하기 바란다.

■ **ParentWnd**
부모 윈도우의 Handle을 지정

■ **FingerWnd**
WindowStyle이 UCBioAPI_WINDOW_STYLE_INVISIBLE로 지정되었을 경우 지문 이미지가 그려질 윈도우의 Handle을 지정한다. 여기에 값을 지정하면 향후 지문 획득 시 지문 이미지가 지정한 윈도우에 표시되며 획득되므로 사용자 정의 획득 UI를 만들 수 있다.

■ **CaptureCallBackInfo**
지문 획득 시 매번 Capture가 이루어 질 때마다 호출될 Callback 함수를 지정한다.

■ **FinishCallBackInfo**
작업이 끝나고 윈도우가 닫히기 직전에 호출될 Callback 함수를 지정한다.

여기서 지정한 Callback을 통해 지문 등록 시에는 지문 등록과 관련된 이벤트 정보도 얻을 수 있다.

■ **CaptionMsg**
지문 등록 중 Cancel 버튼을 눌렀을 때 나오는 메시지박스의 Caption에 표시 될 내용을 지정한다.

■ **CancelMsg**
지문 등록 중 Cancel 버튼을 눌렀을 때 나오는 메시지박스의 취소 안내 메시지 내용을 지정한다.

■ **Option2**
UCBioAPI_WINDOW_OPTION_2 구조체를 통해 추가적으로 다른 옵션을 줄 수 있다. 구조체 멤버에 대한 설명은 다음과 같다. 설정하지 않을 경우 NULL을 지정하면 된다.

◆ **FPForeColor**
지문이 화면에 표시될 색상을 지정 할 수 있다.

◆ **FPBackColor**
지문이 화면에 표시될 때 그 배경색을 지정 할 수 있다.

◆ **DisableFingerForEnroll**
지문 등록 시 등록하지 못하게 할 손가락을 지정 할 수 있다.

각 멤버 값들의 자세한 사용 예는 UCBioBSP SDK를 설치하면 Samples 폴더내의 UCBioBSP_UIDemo 폴더내에서 찾아 볼 수 있다.

3.11.3. Callback 사용하기

UCBioAPI_WINDOW_OPTION 구조체에 지정 가능한 Callback 함수에 대해 설명한다.
사용 가능한 Callback 함수의 정의는 다음과 같다.

```
typedef struct ucbioapi_callback_info_0 {
  UCBioAPI_UINT32               CallBackType;
  UCBioAPI_WINDOW_CALLBACK_0    CallBackFunction;
  UCBioAPI_VOID_PTR             UserCallBackParam;
} UCBioAPI_CALLBACK_INFO_0, *UCBioAPI_CALLBACK_INFO_PTR_0;

typedef struct ucbioapi_callback_info_1 {
  UCBioAPI_UINT32               CallBackType;
  UCBioAPI_WINDOW_CALLBACK_1    CallBackFunction;
```

```
    UCBioAPI_VOID_PTR               UserCallBackParam;
} UCBioAPI_CALLBACK_INFO_1, *UCBioAPI_CALLBACK_INFO_PTR_1;
```

■ **CallBackType**

CallBackFunction의 Type을 설정한다. 이 값이 0이면 Callback 함수의 첫번째 인자가 UCBioAPI_WINDOW_CALLBACK_PARAM_0 구조체의 포인터가 담겨 오게 되고 이 값이 1이면 UCBioAPI_WINDOW_CALLBACK_PARAM_1 구조체의 포인터가 담겨 오게 된다. 현재는 0과 1, 두 가지 Type만 지원한다. 각각의 구조체는 다음과 같다.

```
typedef struct ucbioapi_window_callback_param_0 {
    UCBioAPI_UINT32          dwQuality;
    UCBioAPI_UINT8*          lpImageBuf;
    UCBioAPI_UINT32          dwDeviceError;

    UCBioAPI_UINT32          dwReserved[8];
    UCBioAPI_VOID_PTR        lpReserved;
} UCBioAPI_WINDOW_CALLBACK_PARAM_0;

typedef struct ucbioapi_window_callback_param_1 {
    UCBioAPI_UINT32          dwResult;

    UCBioAPI_UINT32          dwStartTime;
    UCBioAPI_UINT32          dwCapTime;
    UCBioAPI_UINT32          dwEndTime;

    UCBioAPI_UINT32          Reserved[8];
    UCBioAPI_VOID_PTR        lpReserved;
} UCBioAPI_WINDOW_CALLBACK_PARAM_1;
```

UCBioAPI_WINDOW_CALLBACK_PARAM_0의 경우는 UCBioAPI_WINDOW_OPTION의 CaptureCallBackInfo를 위해 사용되고 UCBioAPI_WINDOW_CALLBACK_PARAM_1의 경우는 FinishCallBackInfo를 위해 사용된다. 각각의 멤버에 대한 자세한 설명은 API 레퍼런스를 참조하기 바란다.

■ **CallBackFunction**

호출되어질 Callback 함수를 지정한다. CallBackType에 따른 각각의 함수 정의는 다음과 같다.

```
typedef UCBioAPI_RETURN (WINAPI* UCBioAPI_WINDOW_CALLBACK_0)
            (UCBioAPI_WINDOW_CALLBACK_PARAM_PTR_0, UCBioAPI_VOID_PTR);

typedef UCBioAPI_RETURN (WINAPI* UCBioAPI_WINDOW_CALLBACK_1)
```

```
(UCBioAPI_WINDOW_CALLBACK_PARAM_PTR_1, UCBioAPI_VOID_PTR);
```

두 번째 인자에는 다음에서 지정하는 UserCallBackParam에 지정한 값이 담겨온다.

■ **UserCallBackParam**
Callback 함수의 두 번째 인자로 전달 될 사용자 값을 지정한다.

3.11.4. 사용 예

실제 사용 예는 다음과 같다.

■ **Example**

```
…
UCBioAPI_WINDOW_OPTION winOption;
memset(&winOption, 0, sizeof(UCBioAPI_WINDOW_OPTION));

winOption.Length = sizeof(UCBioAPI_WINDOW_OPTION);
winOption.WindowStyle = UCBioAPI_WINDOW_STYLE_INVISIBLE;
winOption.ParentWnd = this;
winOption.FingerWnd = m_hwndFinger;
…

err = UCBioAPI_Capture(m_hUCBioAPI,
                       UCBioAPI_FIR_PURPOSE_VERIFY,
                       &hCapturedFIR,
                       UCBioAPI_USE_DEFAULT_TIMEOUT,
                       NULL,
                       &winOption);
…
```

좀 더 자세한 사용 예는 UCBioBSP SDK를 설치하면 Samples 폴더내의 UCBioBSP_UIDemo 폴더 내에서 찾아 볼 수 있다.

3.12. Smart Card 사용하기

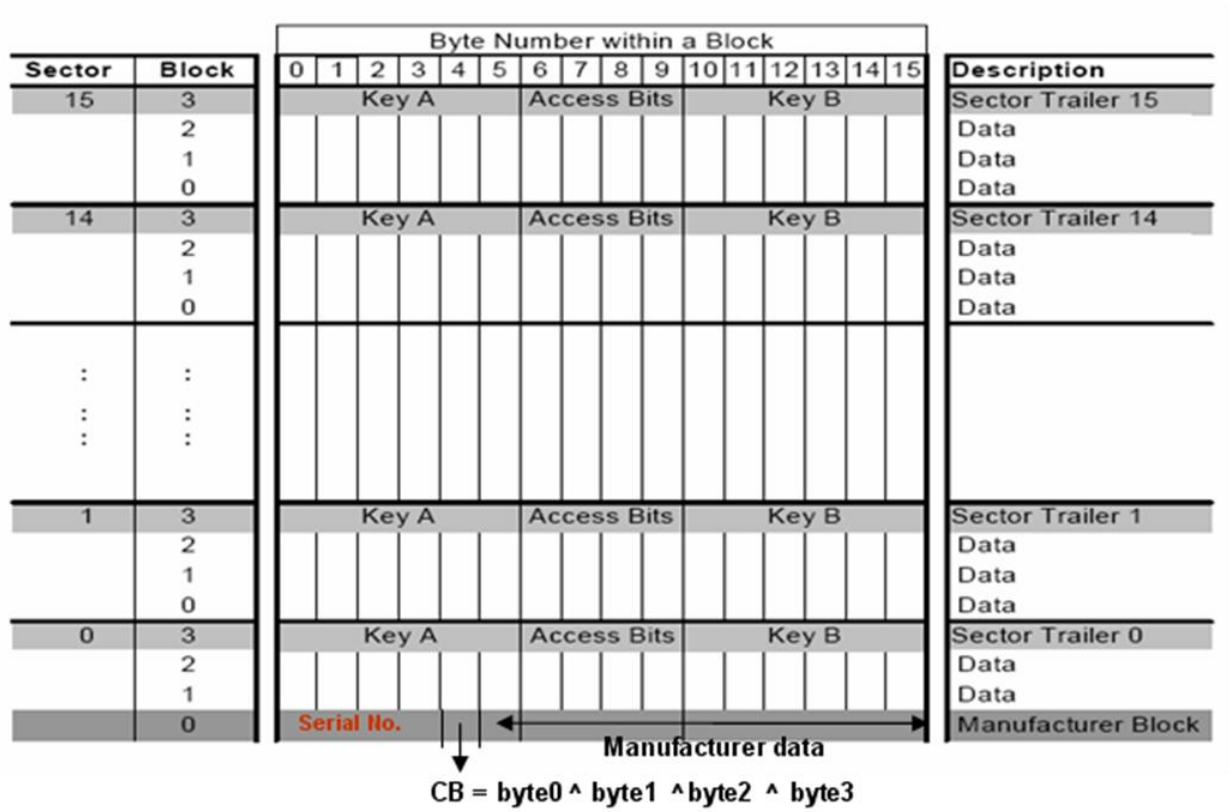
UCBioBSP SDK는 스마트카드를 지원하는 디바이스의 경우 스마트카드를 읽고 쓸 수 있는 기능을 제공한다. 스마트카드를 사용하기 위해서는 반드시 UCBioAPI_OpenDevice 함수를 통해 디바이스를 사용 할 수 있는 상태로 만들어놓은 후 스마트카드 관련 함수들을 호출 하여야 한다.
이 장에서는 스마트카드를 사용하는 방법에 대해 알아본다. 스마트카드 관련 함수들을 사용하기 위해서는 UCBioAPI_SmartCard.h 파일을 include해서 사용해야 한다.

- **참고 - 스마트카드 사용을 위한 함수들은 디바이스의 Firmware 버전에 따라 지원하지 않는 함수가 있을 수 있다.**

3.12.1. 스마트카드의 개요

스마트카드의 전반적인 설명을 이 문서에서 모두 할 수는 없으며 자세한 내용은 관련 문서들을 통해 숙지하도록 한다. 여기에서는 Mifare 카드의 EEPROM의 내부구조와 접근권한 등에 대해서만 간략하게 설명한다.

1) EEPROM의 구조



위 그림은 1Kbytes짜리 Mifare카드의 EEPROM 메모리 구조를 나타낸 그림이다.
그림에서 보면 알 수 있듯이 EEPROM은 16개의 Sector(0~15)와 각 Sector당 4개의 Block(0~3)으

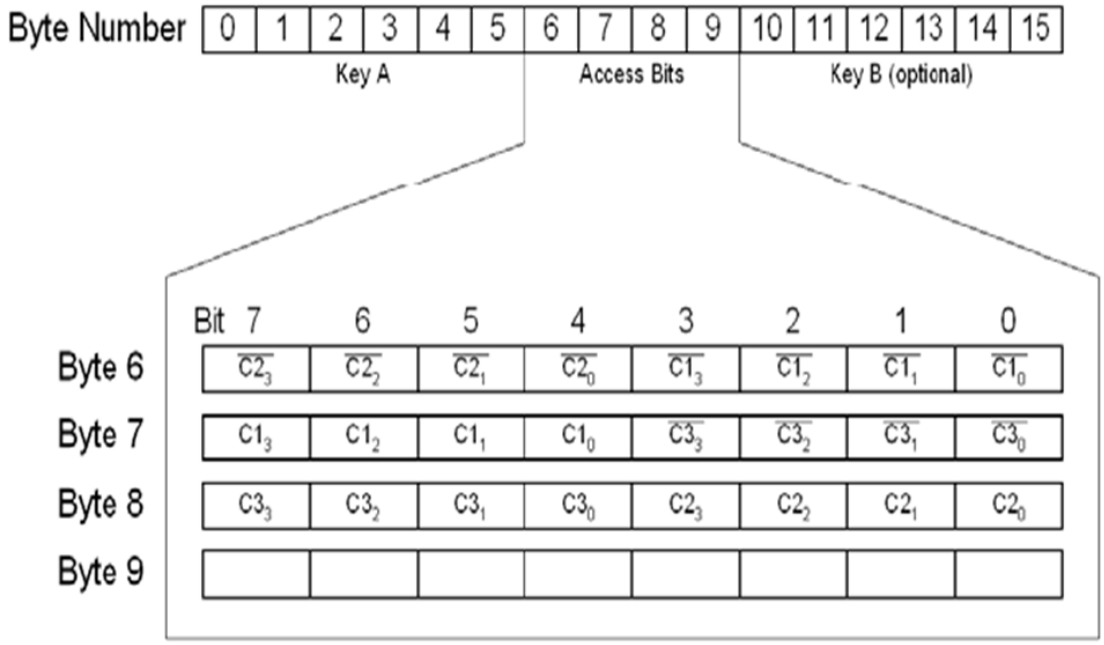
로 구성되어 있다. 즉, 한 Block의 크기가 16bytes이므로 각 Sector는 64bytes를 가지게 되고 이것이 16개가 모여 총 1Kbytes의 공간을 가지게 된다.

```
16 Bytes X 4 Blocks X 16 Sectore = 1024 Bytes
```

각 Sector의 네 번째 Block인 Block3은 각 Sector에 대한 Key값들과 접근 권한을 위한 Access Bits를 가지고 있는 Block으로 Sector Trailer라고 한다. 이 Block에는 일반 데이터를 저장할 수 없으며 이 값에 의해 Sector 내부에 값을 읽거나 쓸 수 있는 권한을 가지게 된다.

2) Access Bits

Sector Trailer의 가운데 4bytes가 그 Sector의 접근 권한을 설정한 값을 담고 있는 Access Bits이다. Access Bits에 대한 값은 다음 그림과 같다.



각 Bits들의 값들 중 아래첨자로 표기된 값은 Block의 위치를 가리킨다. 즉, $C1_3$라고 하면 $_3$은 Block3을 의미한다고 보면 된다. 그렇게 해서 각 Block 별로 C1, C2, C3를 얻을 수 있다. 윗줄이 그어진 값은 Parity Value로 각 C1, C2, C3의 NOT 값이라고 보면 된다.
이 C1, C2, C3를 이용해 각각의 Block에 대한 접근 권한을 지정 할 수 있게 된다. 지정 방법은 Sector Trailer 부분(Block3)과 Data Block(Block0~2)으로 나누어 지며 의미는 아래 표와 같다.

■ **Access conditions for the sector trailer**

Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read
0	1	0	never	never	key A	never	key A	never	Key B may be read
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

■ **Access conditions for the data blocks**

Access bits			Access condition for				Application
C1	C2	C3	read	write	increment	decrement, transfer, restore	
0	0	0	key A|B[1]	key A|B1	key A|B1	key A|B1	transport configuration
0	1	0	key A|B[1]	never	never	never	read/write block
1	0	0	key A|B[1]	key B[1]	never	never	read/write block
1	1	0	key A|B[1]	key B[1]	key B[1]	key A|B[1]	value block
0	0	1	key A|B[1]	never	never	key A|B[1]	value block
0	1	1	key B[1]	key B[1]	never	never	read/write block
1	0	1	key B[1]	never	never	never	read/write block
1	1	1	never	never	never	never	read/write block
Access bits			Access condition for				Application

각각의 값이 의미하는 바는 다음과 같다.

never : 어떠한 방법으로도 사용 할 수 없다.
key A : KeyA를 알고 있다면 사용 할 수 있다.
key B : KeyB를 알고 있다면 사용 할 수 있다.
keyA|B : KeyA 또는 KeyB를 알고 있다면 사용 할 수 있다.

Access Bits에 대한 좀 더 자세한 내용은 Mifare 관련 문서를 참조하기 바란다.

3) Value Block

Data Block을 Value Block으로 변경하게 되면 그 Block은 아래 그림과 같이 4bytes의 값을 저장하는 Block으로 바뀌게 된다. 이렇게 Value Block이 되면 4bytes의 Value값을 써 두고 그 값을 일정 크기로 증가시키거나 감소시킬 수 있는 특수한 용도로 사용된다. Value Block은 데이터 보존 및 보안상 3번에 걸쳐 Block에 기록된다.

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Value				$\overline{\text{Value}}$				Value				Adr	$\overline{\text{Adr}}$	Adr	$\overline{\text{Adr}}$

Value Block으로 사용하기 위해서는 Access Bits의 C1, C2, C3값이 1, 1, 0 또는 0, 0, 1이 되어야 한다. UCBioBSP SDK에서는 이렇게 Value Block으로 변환을 위해 UCBioAPI_SD_PreValue 라는 함수를 제공한다.

3.12.2. 스마트카드의 RF Power 켜기와 끄기

스마트카드를 사용하기 위해서는 우선 스마트카드 Reader기의 RF Power를 켜 주어야 한다. 이렇게 해 주어야지만 스마트카드로부터 데이터를 읽거나 쓰는 작업을 수행 할 수 있다. RF Power를 켜 주기 위해서는 UCBioAPI_SC_RFPowerOn 함수를 호출하면 된다.
스마트카드의 사용을 종료하려면 UCBioAPI_SC_RFPowerOff 함수를 호출하여 Reader기의 RF Power를 꺼 주면 된다.
함수의 인자로는 다음과 같은 사용 할 수 있는데 UCBioAPI_SC_LED_TOGGLE을 지정 할 경우 함수의 성공과 실패 여부를 스마트카드 인식기의 LED에도 붉은색과 파란색으로 나타내 준다.

```
#define UCBioAPI_SC_LED_TOGGLE          (1)
#define UCBioAPI_SC_LED_NOT_TOGGLE      (0)
```

■ **Example - 1**

```
err = UCBioAPI_SC_RFPowerOn(m_hUCBioAPI, UCBioAPI_SC_LED_NOT_TOGGLE);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to RFPowerOn
} else {
   // Failed to RFPowerOn
}
```

■ **Example - 2**

```
err = UCBioAPI_SC_RFPowerOff(m_hUCBioAPI, UCBioAPI_SC_LED_NOT_TOGGLE);
if (err == UCBioAPIERROR_NONE) {
```

```
   // Succeeded to RFPowerOff
} else {
   // Failed to RFPowerOff
}
```

3.12.3. 스마트카드의 Serial 값 읽기

스마트카드에는 카드마다 고유한 Serial 값을 가지고 있다. 이 값을 읽기 위해서는 UCBioAPI_ReadSerial 함수를 사용한다. 이 함수는 내부적으로 UCBioAPI_RFPowerOn 기능을 내장하고 있기 때문에 따로 RF Power를 켜주는 작업을 할 필요가 없으며 Key값을 알 필요도 없다. 간단하게 카드의 Serial 값을 읽기 위해 사용하면 편리하다.

■ **Example**

```
BYTE pSerialBuffer[4] = { 0, 0, 0, 0 };
WORD nSerialBufferLen = 4;
err = UCBioAPI_SC_ReadSerial(m_hUCBioAPI,
                             pSerialBuffer,
                             &nSerialBufferLen,
                             UCBioAPI_SC_LED_TOGGLE);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to read serial
} else {
   // Failed to read serial
}
```

3.12.4. Block 값 읽기 및 쓰기

스마트카드의 EEPROM에 적혀있는 각 Block의 값을 읽고 쓰기 위해서는 반드시 RF Power가 켜져 있어야 한다. 값을 읽기 위해서는 UCBioAPI_SC_ReadBlock 함수를 사용하고 값을 쓰기 위해서는 UCBioAPI_SC_WriteBlock 함수를 사용한다. 인자로는 읽거나 쓰기를 원하는 Sector 번호와 Block 번호를 지정하고 각 Block의 접근권한에 맞는 Key값을 넘겨주면 된다.
사용 예는 다음과 같다.

■ **Example - 1**

```
BYTE pResultBuffer[16];
WORD nResultBufferLen = sizeof(pResultBuffer);
memset(pResultBuffer, 0, sizeof(pResultBuffer));

BYTE pKeyValue[6];
// Set key value to pKeyValue buffer
```

```
err = UCBioAPI_SC_ReadBlock(m_hUCBioAPI,
                            UCBioAPI_SC_USE_KEY_A,
                            nSectorNum, nBlockNum,
                            pKeyValue,
                            pResultBuffer,
                            &nResultBufferLen,
                            UCBioAPI_SC_LED_TOGGLE);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to read block
} else {
   // Failed to read block
}
```

■ **Example - 2**

```
BYTE pDataBuffer[16];
// Set data to pDataBuffer

BYTE pKeyValue[6];
// Set key value to pKeyValue buffer

err = UCBioAPI_SC_WriteBlock(m_hUCBioAPI,
                            UCBioAPI_SC_USE_KEY_A,
                            nSectorNum, nBlockNum,
                            pKeyValue,
                            pDataBuffer,
                            UCBioAPI_SC_LED_TOGGLE);
if (err == UCBioAPIERROR_NONE) {
   // Succeeded to write block
} else {
   // Failed to write block
}
```

이 외의 다른 스마트카드 관련 함수들에 대해서는 API 레퍼런스를 참조하기 바란다.

4.Programming by using COM

이 장에서는 COM을 이용해 프로그래밍 하는 방법에 대해 설명한다.
사용언어는 VB.Net(VS2008) 언어를 기준으로 설명하므로 다른 언어 사용자는 각 언어에 맞도록 적절히 변형하여 사용하여야 한다.

4.1. COM 사용 개요

UCBioBSP SDK에서는 Visual Basic이나 Delphi 등의 RAD(Rapid Application Development) 툴 사용자와 웹 개발을 지원하기 위한 목적으로 COM(Component Object Model)용 DLL을 제공한다. COM용 DLL은 내부적으로 UCBioBSP.dll을 사용하여 모든 처리를 수행하므로 반드시 사용하고자 하는 Application과 동일한 폴더 혹은 시스템 폴더에 존재 해야만 사용이 가능하다.
COM이 UCBioBSP.dll이 제공하는 모든 기능을 다 지원하지는 않지만 COM의 특성을 살려 UCBioBSP.dll에서 제공하지 못하는 편리성도 제공하기 때문에 사용자는 자신에게 맞는 적절한 모듈을 이용해 개발하면 된다.

4.1.1. COM의 등록

COM은 시스템 Registry에 등록이 된 후 사용가능하기 때문에 [Windows키 + R] 버튼을 눌러 나오는 실행 명령 행에 다음과 같이 입력하여 COM 모듈을 시스템에 등록하는 과정이 필요하다.

```
Regsvr32 UCBioBSPCOM.dll
```

이 과정을 거치면 COM 모듈을 사용 할 수 있는 상태가 된다. UCBioBSP SDK를 설치할 경우 32bit용 COM은 이미 설치 과정에서 이루어지므로 따로 등록 할 필요는 없다. 그러나 64bit 용 COM의 경우는 수작업으로 직접 등록해 주어야 한다.

4.2. 초기화 및 종료

COM을 초기화 하고 종료하는 법에 대한 설명을 한다.

4.2.1. 초기화 하기

COM 모듈을 초기화 하는 방법은 새로운 Object로 선언해 주는 방법과 CreateObject 함수를 이용하여 초기화 하는 방법이 있다. 이 두 가지 방법은 동일한 결과를 나타낸다.

■ **Example - 1**

```
Dim WithEvents objUCBioBSP As UCBioBSPCOMLib.UCBioBSP
…
' Create UCBioBSP object
Set objUCBioBSP = New UCBioBSPCOMLib.UCBioBSP
```

이 방법을 사용하려면 UCBioBSPCOM 객체를 사용하기 전에 Visual Basic의 Project 메뉴의 "참조"에서 사용 가능한 참조 항목에 "UNION COMMUNITY – UCBioBSP SDK v3.00 Type Library"를 포함시켜야 한다.

■ **Example - 2**

```
Dim WithEvents objUCBioBSP As UCBioBSPCOMLib.UCBioBSP
…
' Create UCBioBSP object
Set objUCBioBSP = CreateObject("UCBioBSPCOMLib.UCBioBSP")
```

4.2.2. 종료 하기

COM 모듈은 사용이 끝난 개체는 자동으로 해제되기 때문에 개체의 사용 종료를 특별히 알려줄 필요가 없다. 하지만 특정 언어나 환경에서는 명시적으로 종료해 줄 필요가 있는데 다음과 같이 하면 된다.

■ **Example**

```
' Free UCBioBSP object
Set objUCBioBSP = nothing
```

4.2.3. 하위 인터페이스 선언

UCBioBSP SDK의 COM 모듈은 총 7개의 하위 인터페이스가 존재한다. 메인 개체로부터 하위 인터페이스를 얻어올 수 있으며 각각의 고유 기능에 따라 기능이 분류되어 있다. 각각의 인터페이스는 다음과 같다.

1) IDevice

디바이스의 옵션 설정 및 디바이스 Open, Close 등의 디바이스 관련 기능을 수행한다.

```
Dim objDevice As IDevice          ' Declaration Device object
Set objDevice = objUCBioBSP.Device
```

2) IExtraction
지문의 획득, 등록 등의 지문 데이터 추출과 관련된 기능을 수행한다.

```
Dim objExtraction As IExtraction  ' Declaration Extraction object
Set objExtraction = objUCBioBSP.Extraction
```

3) IMatching
지문의 인증과 관련된 기능을 수행한다.

```
Dim objMatching As IMatching      ' Declaration Matching object
Set objMatching = objUCBioBSP.Matching
```

4) IFPData
지문 데이터의 변환등과 같은 데이터 관련 기능을 수행한다.

```
Dim objFPData As IFPData          ' Declaration FPData object
Set objFPData = objUCBioBSP.FPData
```

5) IFPImage
지문 이미지의 추출 및 저장 관련 기능을 수행한다.

```
Dim objFPImage As IFPImage        ' Declaration FPImage object
Set objFPImage = objUCBioBSP.FPImage
```

6) IFastSearch
1:N 엔진인 FastSearch와 관련된 기능을 수행한다.

```
Dim objFastSearch As IFastSearch  ' Declaration FastSearch object
Set objFastSearch = objUCBioBSP.FastSearch
```

7) ISmartCard
스마트카드와 관련된 기능을 수행한다.

```
Dim objSmartCard As ISmartCard    ' Declaration SmartCard object
Set objSmartCard = objUCBioBSP.SmartCard
```

각 하위 인터페이스에 대한 자세한 설명은 COM 레퍼런스를 참조하기 바란다.

4.3. 기본 설정

COM 초기화가 성공하고 나면 SDK를 사용하기 전에 기본적인 설정들을 얻어오거나 새로운 값으로 지정 할 수 있다.

4.3.1. SDK 버전 얻기

현재 사용중인 SDK의 BSP 버전을 얻어 올 수 있다.(버전은 4Step 버전으로 2Step씩 출력됨)
예를 들어 Ver3.4.0.2인 경우 MajorVersion은 "3.4", MinorVersion은 "0.2"가 출력됨

■ **Example**

```
Set objUCBioBSP = New UCBioBSPCOMLib.UCBioBSP
' Check initialize object
If objUCBioBSP.ErrorCode = 0 Then
    ' Get UCBioBSP version
    Caption = objUCBioBSP.MajorVersion & "." & objUCBioBSP.MinorVersion
End If
```

4.3.2. 기본 설정 값 얻어오기 및 새로운 값 설정

SDK의 기본 설정 값을 얻어오거나 새로운 값을 설정 할 수 있다.

■ **Example**

```
txtMaxFinger.Text = objUCBioBSP.MaxFingersForEnroll
txtNecessaryNum.Text = objUCBioBSP.NecessaryEnrollNum
txtSamplesPerFinger.Text = objUCBioBSP.SamplesPerFinger
txtDefaultTimeout.Text = objUCBioBSP.DefaultTimeout
txtEnrollSecuLevel.Text = objUCBioBSP.SecurityLevelForEnroll
txtVerifySecuLevel.Text = objUCBioBSP.SecurityLevelForVerify
txtIdentifySecuLevel.Text = objUCBioBSP.SecurityLevelForIdentify
```

■ **Example**

```
objUCBioBSP.MaxFingersForEnroll = txtMaxFinger.Text
objUCBioBSP.NecessaryEnrollNum = txtNecessaryNum.Text
objUCBioBSP.SamplesPerFinger = txtSamplesPerFinger.Text
objUCBioBSP.DefaultTimeout = txtDefaultTimeout.Text
objUCBioBSP.SecurityLevelForEnroll = txtEnrollSecuLevel.Text
objUCBioBSP.SecurityLevelForVerify = txtVerifySecuLevel.Text
objUCBioBSP.SecurityLevelForIdentify = txtIdentifySecuLevel.Text
```

매인 개체인 UCBioBSP 개체의 Property로서 다음과 같은 값들을 사용 할 수 있다.

■ **MaxFignersForEnroll**

지문 등록을 할 경우 최대 등록 가능하게 할 손가락의 개수를 지정한다.
예를 들어 이 값이 2로 지정될 경우 IExtraction의 Enroll Method를 이용해 지문을 등록 할 경우 최대 2개의 손가락만 등록이 가능하다.
디폴트로 이 값은 10을 가진다.

■ **NecessaryEnrollNum**

지문 등록을 할 경우 최소 등록해야 할 손가락의 개수를 지정한다.
이 값은 반드시 MaxFingersForEnroll 값보다는 작거나 같아야 한다.
예를 들어 이 값이 2로 지정될 경우 IExtraction의 Enroll Method를 이용해 지문을 등록 할 경우 최소 2개의 손가락이 등록되어야지만 등록 과정의 종료가 가능하다.
디폴트로 이 값은 1을 가진다.

■ **SamplesPerFinger**

지문 등록 시 손가락당 몇 개의 Sample이 저장되는지를 지정한다. 현재는 2로 고정되어 있으며 수정 할 수 없다.

■ **DefaultTimeout**

지문 인증 및 등록 시 지문을 획득하기 위해 디바이스가 동작하는 기본 최대 시간을 ms단위로 지정한다. Timeout은 향후 함수 호출 시 따로 지정 할 수 있으나 이때 -1을 지정한 경우 이 값이 사용된다.
디폴트로 이 값은 10000(10초)을 가진다.

■ **SecurityLevelForEnroll / SecurityLevelForVerify / SecurityLevelForIdentify**

지문 등록 / 인증 / 1:N 인증을 위해 사용되어질 인증 보안 레벨을 각각에 대해 설정 할 수 있다. 이 값은 아래와 같은 값을 가질 수 있다.

```
1 - LOWEST
2 - LOWER
3 - LOW
4 - BELOW_NORMAL
5 - NORMAL
6 - ABOVE_NORMAL
7 - HIGH
8 - HIGHER
9 - HIGHEST
```

기본값으로 Enroll/Verify는 5의 값을 가지고 Identify는 6의 값을 가진다.

4.3.3. 입력 지문에 대한 품질 설정

입력 지문에 대한 품질(지문 이미지의 품질)을 설정한다.(항목 3.3.3을 참조바람.)

4.4. 디바이스 사용하기

지문 등록 및 획득을 위해서는 지문 인식 디바이스를 먼저 사용 가능하게 여는 과정이 필요하다. 여기에서는 PC에 설치된 디바이스를 사용하기 위해 필요한 함수들에 대해 설명한다.

4.4.1. 디바이스 목록 얻기

디바이스를 사용하기 전 IDevice의 Enumerate Method를 사용하여 현재 PC에 연결되어 있는 디바이스의 전체 목록 및 개수 등의 정보를 얻어 올 수 있다.

■ **Example**

```
Dim i As Integer

' Enumerate Device
Call objDevice.Enumerate

If objUCBioBSP.ErrorCode <> 0 Then
    comboDevice.AddItem "Auto_Detect"
    For i = 0 To objDevice.EnumCount - 1
        nNameID = objDevice.EnumDeviceNameID(i)
        If nNameID <> 0 Then
            comboDevice.AddItem objDevice.EnumDeviceName(i)
        End If
    Next i
End If
```

IDevice의 Enumerate Method를 사용하고 나면 IDevice의 EnumCount Property에 시스템에 연결되어 있는 디바이스의 전체 개수가 담겨있게 되고 각종 디바이스에 대한 정보가 여러 Property에 배열로 담겨지게 된다. Enumerate Method 호출 후 얻어 올 수 있는 Property의 종류는 다음과 같다.

■ **EnumDeviceID**

디바이스 ID의 목록이 담겨있다. DeviceID는 DeviceNameID와 DeviceInstance가 합해진 값이다.

■ **EnumDeviceNameID**

디바이스 이름에 대한 ID 목록이 담겨있다.

■ **EnumDeviceInstance**

디바이스의 Instance 번호에 대한 목록이 담겨있다. 동일 디바이스가 여러 개 달릴 경우 이 Instance의 값이 늘어나게 된다. 현재는 0만 지원한다.

■ **EnumDeviceDescription**
디바이스의 설명에 대한 목록이 담겨있다.

■ **EnumDeviceDll**
디바이스가 사용하는 DLL 파일명에 대한 목록이 담겨 있다.

■ **EnumDeviceSys**
디바이스가 사용하는 Sys 파일명에 대한 목록이 담겨 있다.

■ **EnumDeviceAutoOn**
디바이스가 AutoOn 기능을 지원하는지에 대한 목록이 담겨있다.

■ **EnumDeviceBrightness / EnumDeviceContrast / EnumDeviceGain**
디바이스의 밝기값 / 대비값 / Gain에 대한 값의 목록이 담겨있다.

4.4.2. 디바이스 열기

디바이스를 사용하기 전에는 IDevice의 Open Method를 이용해 반드시 사용할 디바이스를 Open 해 주어야 한다. 특정 디바이스를 Open하기 위해서는 인자로 Open하고자 하는 디바이스의 ID를 지정해 주면 된다. 만약 최근에 사용한 디바이스를 자동으로 Open하고자 할 경우에는 디바이스 ID에 255를 지정하면 된다. (Visual Basic에서는 &HFF로 지정하면 된다.)

■ **Example**

```
Dim objUCBioBSP As UCBioBSPCOMLib.UCBioBSP
Dim objDevice As IDevice          ' Declaration Device object
…
' Create UCBioBSP object
Set objUCBioBSP = New UCBioBSPCOMLib.UCBioBSP
Set objDevice = objUCBioBSP.Device
…
' Open device
Call objDevice.Open(&HFF) ' &HFF = 0xFF = 255 = Auto

If objUCBioBSP.ErrorCode = 0 Then
   ' Succeeded to open device
Else
   ' Failed to open device
End If
```

디바이스 ID로 사용 할 수 있는 값들은 아래와 같다.

```
1 -  FOH01
2 -  FOM01
3 -  FOH03
4 -  HAM500
5 -  FOH01A
6 -  FOM01A
7 -  FPR02
8 -  FSH01RF
9 -  FOH01RF
10 - FR100
11 - FPR02LFD
12 - FOH01RFL
13 - FSH01SC
14 - FPR02_V30
255- AUTO DETECT
```

4.4.3. 디바이스 닫기

디바이스의 사용이 모두 끝난 뒤에는 IDevice의 Close Method를 이용해 디바이스를 사용 종료를 해 주어야 한다. 이때에는 반드시 처음에 Open한 디바이스를 닫아주어야 한다.
또한 다른 디바이스를 Open하기 위해서는 현재 사용하고 있는 디바이스를 우선 닫아주어야 새로운 디바이스를 Open 할 수 있다.

■ **Example**

```
' Close Device if before opened
Call objDevice.Close(objDevice.OpenedDeviceID)

If objUCBioBSP.ErrorCode = 0 Then
   ' Succeeded to close device
Else
   ' Failed to close device
End If
```

4.4.4. 디바이스 정보 얻기

현재 Open한 디바이스에 대한 정보를 얻고자 할 경우 IDevice의 여러 Property를 이용해 디바이스가 지원하는 이미지의 크기 등과 같은 값을 얻을 수 있다.

■ **Example**

```
…
iDeviceID = &HFF
```

```
' Open device
Call objDevice.Open(iDeviceID)

If objUCBioBSP.ErrorCode = 0 Then
    txtWidth.Text = objDevice.ImageWidth(iDeviceID)
    txtHeight.Text = objDevice.ImageHeight(iDeviceID)
End If
```

4.5. FIR 데이터의 이해

지문을 등록하고 인증하기 위해서는 반드시 UCBioBSP SDK가 사용하는 지문 데이터인 FIR에 대한 이해가 필요하다. 1장에서 지문 데이터의 구조에 대해서는 설명을 했지만 실제 COM에서 FIR을 사용하기 위해서 알아야 할 것들에 대해 설명한다.

4.5.1. FIR의 종류

COM에서 사용하는 FIR은 다음의 두 가지 종류로 구분 될 수 있다.

■ **Binary FIR**
지문 데이터에 대한 암호화된 바이너리 메모리 블록을 포함한 값이다. 이 값을 얻기 위해서는 IExtraction의 FIR Property를 통해 얻을 수 있다.

■ **String FIR**
지문 데이터에 대한 암호화된 문자열 형태의 값이다. 이 값을 얻기 위해서는 IExtraction의 TextFIR Property를 통해 얻을 수 있다.

두 가지 데이터 중 어떤 것을 사용하나 동일하며 String FIR의 경우는 문자열로 이루어져 있기 때문에 좀 더 사용하기 쉬운 특징을 가지고 있다.

4.5.2. FIR의 사용

인증 등의 함수에서 FIR을 사용하기 위해서는 FIR의 종류에 구분 없이 Binary FIR이나 String FIR로 된 데이터를 Method의 인자로 넘기기만 하면 된다.

FIR 종류에 따른 각각의 사용법은 다음과 같다.

■ **Example**

```
' Verify
If radioBinaryFIR.Value Then
    Call objMatching.Verify(binaryEnrolledFIR)  ' Verify with binary FIR
Else
    Call objMatching.Verify(szTextEnrolledFIR)  ' Verify with String FIR
End If
```

4.5.3. FIR 메모리 해제

COM 모듈은 사용이 끝난 개체는 자동으로 해제되기 때문에 개체의 메모리를 특별히 해 줄 필요가 없다.

4.6. 지문 등록하기

지문을 등록하기 위해서는 반드시 먼저 디바이스가 Open되어 있어야 한다.
COM 모듈에서는 지문을 등록하기 위해서는 IExtraction 인터페이스를 사용하여야 한다.
지문을 등록하게 되면 IExtraction의 FIR 또는 TextFIR Property에는 지문 데이터가 담겨지게 된다.

4.6.1. 신규 지문 등록 및 기존 지문 수정

지문을 등록하기 위해서는 IExtraction의 Enroll 함수를 사용한다. 지문 등록 함수는 등록뿐만 아니라 이미 기존에 등록된 지문을 수정 할 수 있는 기능도 가지고 있다. 지문 수정을 위해서는 두 번째 인자에 기존 지문을 지정하면 된다. 새로운 지문을 등록할 경우에는 이 값을 Null로 지정하면 된다.

4.6.2. Payload 지정

지문 등록 시 Payload를 지정해 등록된 지문 데이터 내부에 사용자의 특정 데이터를 암호화해서 삽입 할 수도 있다. 이렇게 등록된 Payload는 지문 인증이 성공 할 경우에만 얻어 올 수 있다.

4.6.3. 자동 지문 감지(Auto Detect)

지문 입력장치의 기본 기능은 지문을 입력받을 때 LED를 켬으로서 지문입력을 유도하게 된다. 그러나 이 기능을 사용하지 않고 지문을 입력장치에 가져다 댄 경우 자동으로 지문을 획득하기 위한 방식으로 모든 지문입력 명령 전에 지시하면 된다.

4.6.4. 사용 예

아래 예제에서는 기존 등록 데이터를 수정하면서 새로운 Payload 데이터로 바꾸는 것을 알 수 있다. 좀 더 자세한 예제는 SDK 설치 후 설치 폴더내의 Samples 폴더내의 UCBioBSPCOM_DemoVB 폴더에서 찾아 볼 수 있다.

■ **Example**

```
Dim binaryEnrolledFIR() As Byte
Dim szTextEnrolledFIR As String
Dim szPayload As String

' Get Payload
szPayload = txtPayload.Text

' Enroll
Call objExtraction.Enroll(szPayload, szTextEnrolledFIR)

If objUCBioBSP.ErrorCode = 0 Then
```

```
   ' Get binary encoded FIR data
   binaryEnrolledFIR = objExtraction.FIR

   ' Get text encoded FIR data
   szTextEnrolledFIR = objExtraction.TextFIR

   labelStatus.Caption = "Succeeded to enroll."
Else
   labelStatus.Caption = "Failed to enroll."
End If

' AutoDetect 기능 사용예제
Call objDevice.Open(&HFF) ' 장비 Open
Call objDevice.SetAutoDetect(0) ' 기존 방식으로 사용
Call objDevice.SetAutoDetect(1) ' AutoDetect 사용
```

4.7. 지문 획득하기

지문을 획득하기 위해서는 반드시 먼저 디바이스가 Open되어 있어야 한다.
COM 모듈에서는 지문을 획득하기 위해서는 IExtraction 인터페이스를 사용하여야 한다.
지문을 획득하는 것은 지문 등록과 마찬가지로 획득한 지문 데이터를 FIR로 얻게 되지만 등록과는 달리 단지 하나의 라이브 지문만을 입력 받는다.
이렇게 획득된 지문 데이터와 이전에 등록 받은 지문 데이터를 비교하여 지문 인증을 수행 할 수 있다.

4.7.1. 지문 획득

지문을 획득하기 위해서는 IExtraction의 Capture Method를 사용한다. 획득을 위한 목적을 지정할 수 있는데 사용 할 수 있는 값으로는 다음과 같다.

```
1 -  VERIFY
2 -  IDENTIFY
3 -  ENROLL
4 -  ENROLL_FOR_VERIFICATION_ONLY
5 -  ENROOL_FOR_IDENTIFICATION_ONLY
6 -  AUDIT
16 - UPDATE
```

만약 등록용 목적을 지정 할 경우 Enroll Method를 부르는 것과 동일한 효과를 가진다.
지정된 목적은 참고용으로만 사용되며 향후 인증에 영향을 미치지는 않는다.

4.7.2. 사용 예

■ **Example**

```
Dim binaryEnrolledFIR() As Byte
Dim szTextEnrolledFIR As String

' Capture
Call objExtraction.Capture(1)

If objUCBioBSP.ErrorCode = 0 Then
   ' Get binary encoded FIR data
   binaryEnrolledFIR = objExtraction.FIR

   ' Get text encoded FIR data
   szTextEnrolledFIR = objExtraction.TextFIR

   labelStatus.Caption = "Succeeded to capture."
```

```
Else
   labelStatus.Caption = "Failed to capture."
End If
```

4.8. 지문 인증하기

COM 모듈에서는 지문을 인증하기 위해서는 IMatching 인터페이스를 사용하여야 한다.
지문을 인증하기 위해서는 반드시 디바이스가 있어야 하는 것은 아니지만 라이브 지문과의 인증을 위해서는 반드시 먼저 디바이스가 Open 되어 있어야 한다.
지문 인증을 위해서는 크게 아래 두 가지 방식이 사용된다.

4.8.1. 라이브 지문과 등록 지문과의 인증

이미 등록 받은 지문 데이터를 입력 값으로 주고 현재 지문인식기로부터 지문을 라이브로 입력 받아 서로 비교하는 방식이다. 때문에 반드시 디바이스가 Open된 상태로 사용하여야 한다.
이 방식으로 인증하기 위해서는 IMatching의 Verify Method를 사용한다.

■ **Example**

```
' Verify
If radioBinaryFIR.Value Then
    Call objMatching.Verify(binaryEnrolledFIR)  ' Verify with binary FIR
Else
    Call objMatching.Verify(szTextEnrolledFIR)  ' Verify with String FIR
End If

If objUCBioBSP.ErrorCode <> 0 Then
    labStatus.Caption = "Verify Function Failed !"
    Exit Sub
End If

' Check result of verify
If objMatching.MatchingResult Then
    ' Check payload
    If objMatching.ExistPayload Then
        labelStatus.Caption = objMatching.TextPayload
    Else
        labelStatus.Caption = "Verify Succeeded!"
    End If
Else
    ' Show fail message.
    labStatus.Caption = "Verify Failed!"
End If
```

4.8.2. 미리 획득한 지문과 등록 지문과의 인증

이미 등록 받은 지문 데이터와 미리 획득한 지문 데이터를 입력 값으로 주고 그 두 데이터를 비

교하는 방식이다. 때문에 디바이스가 없어도 사용이 가능하다. 서버에서 클라이언트로부터 전송되어져 온 지문 데이터의 인증만을 할 경우 사용하는 방식이다.
이 방식으로 인증하기 위해서는 IMatching의 VerifyMatch Method를 사용한다.

■ **Example**

```
Dim szTextEnrolledFIR As String
Dim szTextCapturedFIR As String

' Enroll
Call objExtraction.Enroll(Null, Null)

If objUCBioBSP.ErrorCode = 0 Then
   ' Get text encoded FIR data
   szTextEnrolledFIR = objExtraction.TextFIR
End If

' Capture
Call objExtraction.Capture(1)

If objUCBioBSP.ErrorCode = 0 Then
   ' Get text encoded FIR data
   szTextCapturedFIR = objExtraction.TextFIR
End If

' VerifyMatch
Call objMatching.VerifyMatch(szCapturedFIR, szTextEnrolledFIR)

If objUCBioBSP.ErrorCode <> 0 Then
    labStatus.Caption = "VerifyMatch Function Failed !"
    Exit Sub
End If

' Check result of verify
If objMatching.MatchingResult Then
    ' Check payload
    If objMatching.ExistPayload Then
        labelStatus.Caption = objMatching.TextPayload
    Else
        labelStatus.Caption = "Verify Succeeded!"
    End If
Else
```

```
    ' Show fail message.
    labStatus.Caption = "Verify Failed!"
End If
```

4.8.3. Payload 데이터 얻기

지문을 인증이 성공하면 지문 등록 시 삽입했던 Payload를 얻을 수 있다. Payload 데이터는 어떠한 데이터도 사용 될 수 있기 때문에 사용자 인증 시 특정 고정된 값을 얻고자 할 경우 사용 할 수 있다.

4.9. FastSearch (1:N 인증) 사용하기

UCBioBSP SDK는 1:N의 고속 인증을 위해 FastSearch Engine을 제공한다. 많은 수의 사용자를 인증할 경우 단순히 1:1 인증의 반복을 통해서는 효과적인 인증 속도를 기대하기 어렵다. 때문에 1:N만을 위한 인증이 필요하게 되고 FastSearch Engine을 통해 다수의 사용자 중 1명을 인증하는 기능을 제공하게 되었다. COM 모듈에서는 FastSearch 엔진을 사용하기 위해서는 IFastSearch 인터페이스를 사용하여야 한다.

4.9.1. 초기화 및 종료

COM에서는 FastSearch Engine을 사용하기 위해 특별한 초기화나 종료를 할 필요가 없다. 단지 IFastSearch 인터페이스를 메인 개체로부터 가져오기만 하면 내부적으로 초기화가 이루어져 사용 가능한 상태가 된다.

■ **Example**

```
Dim objUCBioBSP As UCBioBSPCOMLib.UCBioBSP
Dim objFastSearch As IFastSearch        ' Declaration FastSearch object
…
' Create UCBioBSP object
Set objUCBioBSP = New UCBioBSPCOMLib.UCBioBSP
Set objFastSearch = objUCBioBSP.FastSearch
…
```

4.9.2. 기본 설정 값 얻어오기 및 새로운 값 설정

FastSearch Engine의 기본 설정 값을 얻어오거나 새로운 값을 설정 할 수 있다.
기본 설정값으로 사용되는 Property는 다음과 같은 것들이 있다.

■ **MaxSearchTime**

Identify를 수행할 최대 시간을 지정한다. 단위는 millisecond 단위이다. 만약 10,000이라고 지정하면 10초간 Identify를 수행했는데도 인증이 되지 않았다면 Identify를 종료하게 된다. 이 값을 0으로 지정하면 시간 제한 없이 Identify를 수행하게 된다. 기본값은 0이다.

■ **UseGroupMatch**

인증을 수행할 때 그룹단위의 인증을 할 것인지를 결정한다. 0일 경우 DB에 들어있는 순서대로 인증을 수행하고 1일 경우 그룹단위의 인증을 수행한다. 기본값은 1이다.
이 값은 가능한 1로 설정하고 인증을 하는 것이 좋다.

■ **MatchMethod**

인증을 수행할 방법을 결정한다. 0일 경우 설정된 인증 레벨을 이용해 그 레벨을 넘는 것이 나오면 바로 인증을 종료하는 방법이고 1일 경우 최고점 인증 방법으로 가장 인증 레벨이

높은 값을 찾는 방식이다. 기본값은 0이다. 이 값은 가능한 0으로 설정하고 인증을 하는 것이 좋다.

4.9.3. DB 만들기

1:N 인증을 수행하기 위해서는 우선 인증을 수행할 다수의 DB를 메모리상에 먼저 만들어야 한다. 고속의 인증을 위해 내부적으로 메모리 DB를 만들어 인증을 수행하게 된다. 그러기 위해서는 각각의 FIR 데이터를 하나의 인증용 메모리 DB로 묶는 작업이 필요한데 IFastSearch의 AddFIR Method는 입력 받은 FIR 데이터나 사용자 DB에 등록되어 있는 FIR 데이터를 하나의 메모리 DB로 만들어 준다.
또한 FastSearch Engine은 내부적으로 FIR을 하나의 데이터 단위로 사용하지 않고 Template 단위를 데이터 단위로 사용한다. 때문에 만약 FIR 내부에 여러 개의 Template이 포함된 경우라면 하나의 FIR을 DB에 추가하더라도 내부적으로 여러 개의 Template이 DB에 추가되어질 수 있다.
입력값으로는 DB에 등록할 FIR 데이터와 그 데이터에 대한 사용자 ID 값을 넘겨준다. 향후 1:N 인증이 성공하면 이때 넘겨준 사용자 ID 값을 얻을 수 있다.

■ **Example**

```
Dim nUserID As Long
Dim szFir As String
…
' Set User ID
nUserID = 1

' Get FIR data
Call objExtraction.Enroll(Null)
If objUCBioBSP.ErrorCode = 0 Then
    szFir = objExtraction.TextFIR

    ' Regist FIR to FastSearch DB
    Call objFastSearch.AddFIR(szFir, nUserID)
    If objUCBioBSP.ErrorCode = 0 Then
        MsgBox "Succeeded to add FIR to DB"
    Else
        MsgBox objUCBioBSP.ErrorDescription
    End If
Else
    MsgBox objUCBioBSP.ErrorDescription
End If
…
```

4.9.4. 메모리 DB 관리

만들어진 인증용 메모리 DB를 관리하기 위해 다음과 같은 다양한 Method가 제공된다.

■ **RemoveFp**

메모리 DB에서 특정 지문 하나를 삭제한다.

■ **RemoveUser**

메모리 DB에서 특정 사용자의 지문을 모두 삭제한다. 메모리 DB에는 특정 사용자에 대해 다수의 지문 정보가 존재 할 수 있으므로 이것을 한꺼번에 삭제 하고자 할 경우 유용하다.

■ **ClearDB**

메모리 DB 전체를 지운다.

■ **FpCount**

메모리 DB에 있는 지문의 개수를 얻어온다.

■ **FpInfo**

메모리 DB에 있는 특정 위치의 지문 정보를 얻어온다.

■ **AddedFpCount**

직전에 DB에 추가한 FIR의 Template 개수를 얻어온다.

■ **AddedFpInfo**

직전에 DB에 추가한 FIR의 Template 정보를 얻어온다.

■ **IsFpExist**

특정 지문 정보가 메모리 DB에 있는지를 검사한다.

■ **SaveDBToFile**

메모리 DB 전체를 파일로 저장한다. 이렇게 저장해둔 DB는 LoadDBFromFile Method를 이용해 다시 읽을 수 있다.

■ **LoadDBFromFile**

File로 저장된 DB를 메모리 DB로 다시 읽어 들인다.
이렇게 할 경우 AddFIR Method를 이용해 새로 DB를 구축하는 것 보다 훨씬 빠르게 메모리 DB를 만들 수가 있다.

4.9.5. 1:N 인증하기

만들어진 메모리 지문 DB에서 특정 사용자의 지문을 인증하기 위해서는 IFastSearch 인터페이스의 IdentifyUser Method를 사용하면 된다.

인증 시에는 인증을 위한 보안 레벨 값을 정할 수 있다.

■ **Example**

```
…
Dim i As Integer
Dim szTextFIR As String
Dim ListItem As ListItem
Dim objMatchedFpInfo As ITemplateInfo

' Set MaxTime for IdentifyUser
objFastSearch.MaxSearchTime = 5000  ' Set Maxtime to 5 seconds.

nUserID = 0
szTextFIR = ""

' Get FIR data
Call objDevice.Open(&HFF)    ' 0xFF : Auto detect to device ID
Call objExtraction.Capture(2)   ' 2 : Capture for identify purpose
If objUCBioBSP.ErrorCode = 0 Then
    szTextFIR = objExtraction.TextFIR

    ' Identify FIR to IndexSearch DB
    Call objFastSearch.IdentifyUser(szTextFIR, 5)

    If objUCBioBSP.ErrorCode = 0 Then
        Set objMatchedFpInfo = objFastSearch.MatchedFpInfo
        If objUCBioBSP.ErrorCode = 0 Then
            ' Add item to list of result
            Set ListItem = ListResult.ListItems.Add
            ListItem.Text = objMatchedFpInfo.UserID
            ListItem.SubItems(1) = objMatchedFpInfo.FingerID
            ListItem.SubItems(2) = objMatchedFpInfo.SampleNumber
            Set ListItem = Nothing
        End If
    Else
        MsgBox objUCBioBSP.ErrorDescription
    End If
Else
    MsgBox objUCBioBSP.ErrorDescription
End If
```

```
Call objDevice.Close(&HFF)   ' 0xFF : Auto detect to device ID
…
```

FastSearch에 대한 자세한 사용 예는 SDK 설치 후 Samples 폴더내의 UCBioBSPCOM_FastSearchDemoVB 폴더에서 찾을 수 있다.

4.10. FIR 데이터 변환하기

앞에서도 언급했듯이 FIR 데이터는 다수의 Template 데이터로 구성된 데이터의 집합이다. 때문에 FIR 데이터로부터 각각의 Template 데이터를 독립적으로 얻고자 할 경우나 또는 다수의 Template 데이터를 이용해 하나의 FIR 데이터를 만들고자 할 경우에는 변환 함수를 사용해야 한다.
더불어 이미지를 담는 Audit FIR의 경우에도 제공되는 변환 함수들을 이용하여 각각의 손가락에 대한 Raw 이미지를 얻을 수 있다.
COM 모듈에서는 변환 관련 기능을 사용하기 위해서는 IFPData와 IFPImage 인터페이스를 사용하여 한다.

4.10.1. FIR 데이터로부터 Template 데이터 추출

FIR 데이터로부터 Template 데이터를 추출하기 위해서는 IFPData의 Export Method를 사용한다. 이 Method를 호출하고 나면 Template 데이터뿐만 아니라 FIR의 각종 정보들도 얻을 수 있다.

■ **Example**

```
Dim biFIR() As Byte
Dim biTemplate() As Byte
Dim nTemplateType As Integer
nTemplateType = 400
…
objExtraction.Capture 1
…
' Get binary encoded FIR data
biFIR = objExtraction.FIR

' Export data
objFPData.Export biFIR, nTemplateType
If objUCBioBSP.ErrorCode = 0 Then
    ' Get template data
    nFingerID = objFPData.FingerID(0)
    biTemplate = objFPData.FPData(nFingerID, 0)
Else
    ' Failed to export data
End If
```

4.10.2. Template 데이터를 이용해 FIR 만들기

Template 데이터로부터 FIR을 만들기 위해서는 IFData의 Import Method를 사용한다. Import Method를 여러 번 호출하여 여러 개의 Template이 들어간 하나의 FIR을 만들고자 할 경우에는 첫 번째 인자를 0으로 지정한 후 원하는 횟수만큼 Import Method를 호출하면 된다.

만약 이 값을 1로 지정하면 이전에 Import했던 데이터는 지우고 새로 FIR을 만들게 된다.

■ **Example**

```
Dim biTemplate() As Byte
Dim nDataSize As Long
Dim nTemplateType As Integer
nTemplateType = 400
…
' Set template data to biTemplate buffer
…
' Import data
objFPData.Import 1, nFingerID, 1, nTemplateType, nDataSize, biTemplate
If objUCBioBSP.ErrorCode = 0 Then
    ' Verify
    objDevice.Open &HFF ' Auto
    objMatching.Verify objFPData.FIR
    objDevice.Close &HFF

    If objMatching.MatchingResult Then
        ' Succeeded to verify!
    Else
        ' Failed to verify!
    End If
Else
    ' Failed to import data
End If
```

4.10.3. Audit FIR 데이터로부터 Raw 이미지 추출

Audit FIR은 IExtraction 인터페이스의 Capture나 Enroll Method 사용 시 얻어지는 이미지 정보를 담고 있는 FIR 데이터이다. 이 Audit FIR은 일반 FIR과 동일한 구조를 가지지만 내부적으로 이미지를 담고 있다는 것만 다르다. 이 Audit FIR 데이터를 Raw 이미지로 추출하기 위해서는 IFPImage의 Export Method를 사용하면 된다.
또한 이렇게 얻어진 Raw 이미지는 IFPImage의 Save와 같은 Method를 이용해 다양한 포맷의 이미지 파일로도 저장이 가능하다. 사용 가능한 이미지 포맷은 다음과 같다.

```
1 -  RAW
2 -  BMP
3 -  JPG
4 -  WSQ
```

■ **Example**

```
Dim nImgType As Integer
nImgType = 2  ' BMP
…
objExtraction.Capture 1
…
' Export image data
objFPImage.Export
If objUCBioBSP.ErrorCode = 0 Then
    ' Save image data to file
    nFingerID = objFPData.FingerID(0)
    objFPImage.Save szFileName, nImgType, nFingerID
Else
    ' Failed to export data
End If
```

4.11. UI 설정하기

UCBioBSP SDK에서 사용되는 사용자 인터페이스를 구성하기 위한 방법에 대해 설명한다.
COM 모듈에서는 UI 관련 Property를 설정하기 위해서는 UCBioBSPCOM의 메인 인터페이스를 사용하여야 한다.

4.11.1. Skin 파일 불러오기

UCBioBSP SDK에서는 사용하는 등록 및 인증용 화면을 Skin 방식의 UI로 사용하고 있다. 때문에 UCBioBSP SDK에서 기본 제공하는 UI를 사용하지 않고 다른 언어로 된 UI나 또는 다른 형태의 UI를 사용하고자 하는 경우에는 사용자 정의 Skin을 만들어 사용 할 수 있다. 이때 만들어진 Skin DLL을 읽어 사용하는 Method가 SetSkinResource이다.
현재 UCBioBSP SDK는 기본적으로 영문으로 된 Skin을 내장하고 있다. 만약 이것을 한글로 된 Skin으로 바꾸고자 할 경우에는 다음과 같이 하면 된다.

■ **Example**

```
' Set skin resource
Call objUCBioBSP.SetSkinResource(szSkinFileName)
If objUCBioBSP.ErrorCode = 0 Then
    // Succeeded to change to new skin
Else
   // Failed to change to new skin
End If
```

사용자 정의 Skin을 만들기 위해서는 본사로 문의하면 된다.

4.11.2. UI 속성 변경하기

UCBioBSP SDK에서는 사용자가 임의로 UI 관련 속성을 변경해 작업 할 수 있는 기능을 제공하고 있다. COM에서는 설정 값을 Property로 제공하고 있으면 각각의 값은 다음과 같다.

■ **WindowStyle**

윈도우가 화면에 표시되는 형태를 지정 할 수 있다. 윈도우가 팝업 형태로 뜰 것인지 아니면 다른 윈도우의 영역에 지문만 표시 할 것인지를 지정 할 수 있다.

```
0 -  POPUP
1 -  INVISIBLE
```

■ **WindowOption**

윈도우가 화면에 표시되는 형태에 대한 Flag를 지정 할 수 있다. 지문이 화면에 표시되지 않게 하거나 지문 등록 시 Welcome 페이지를 없애는 등의 설정을 할 수 있다. 이 세가지 Flag

는 중복해서 사용 가능하므로 OR 연산자를 이용해 지정 가능하다.

```
0x00010000 – NO_FPIMG
0x00020000 – NO_WELCOME
0x00040000 – NO_TOPMOST
```

각 Flag에 대한 자세한 설명은 API 레퍼런스를 참조하기 바란다.

■ **ParentWnd**
부모 윈도우의 Handle을 지정

■ **FingerWnd**
WindowStyle이 INVISIBLE(1)로 지정되었을 경우 지문 이미지가 그려질 윈도우의 Handle을 지정한다. 여기에 값을 지정하면 향후 지문 획득 시 지문 이미지가 지정한 윈도우에 표시되며 획득되므로 사용자 정의 획득 UI를 만들 수 있다.

■ **CaptionMsg**
지문 등록 중 Cancel 버튼을 눌렀을 때 나오는 메시지박스의 Caption에 표시 될 내용을 지정한다.

■ **CancelMsg**
지문 등록 중 Cancel 버튼을 눌렀을 때 나오는 메시지박스의 취소 안내 메시지 내용을 지정한다.

■ **FPForeColor**
지문이 화면에 표시될 색상을 지정 할 수 있다.

■ **FPBackColor**
지문이 화면에 표시될 때 그 배경색을 지정 할 수 있다.

■ **DisableFingerForEnroll**
지문 등록 시 등록하지 못하게 할 손가락을 지정 할 수 있다.

각 멤버 값들의 자세한 사용 예는 UCBioBSP SDK를 설치하면 Samples 폴더내의 UCBioBSPCOM_UIDemoVB 폴더내에서 찾아 볼 수 있다.

4.11.3. Callback Event Handler사용하기

COM용 개체를 선언 할 때 다음과 같이 WithEvents를 지정하고 선언 할 경우 COM 개체로 부터 이벤트를 받을 수 있다.

```
' Declaration global variables
Dim WithEvents objUCBioBSP As UCBioBSPCOMLib.UCBioBSP
```

받을 수 있는 이벤트는 Capture 할 때마다 불리는 Capture 이벤트와 Enroll Method를 수행 할 때 불리는 Enroll 이벤트가 있다. 각각의 Event Handler로부터 필요한 값들을 얻을 수가 있다.

■ **Example**

```
' Capture event handler
Private Sub objUCBioBSP_OnCaptureEvent(ByVal Quality As Long)
    …
End Sub

' Enroll event handler
Private Sub objUCBioBSP_OnEnrollEvent(ByVal EventID As Long)
    …
End Sub
```

좀 더 자세한 사용 예는 UCBioBSP SDK를 설치하면 Samples 폴더내의 UCBioBSP_UIDemo 폴더 내에서 찾아 볼 수 있다.

4.12. Smart Card 사용하기

UCBioBSP SDK는 스마트카드를 지원하는 디바이스의 경우 스마트카드를 읽고 쓸 수 있는 기능을 제공한다. 스마트카드를 사용하기 위해서는 반드시 IDevice.Open Method를 통해 디바이스를 사용할 수 있는 상태로 만들어놓은 후 스마트카드 관련 기능들을 호출 하여야 한다.
이 장에서는 스마트카드를 사용하는 방법에 대해 알아본다. COM 개체에서 스마트카드 관련 기능들을 사용하기 위해서는 ISmartCard 인터페이스를 사용해야 한다.

- **참고 - 스마트카드 사용을 위한 함수들은 디바이스의 Firmware 버전에 따라 지원하지 않는 Method가 있을 수 있다.**

스마트카드의 개요는 3.12.1. 장에 설명된 내용을 참조하기 바란다.

4.12.1. 초기화와 종료

COM에서는 SmartCard을 사용하기 위해 특별한 초기화나 종료를 할 필요가 없다. 단지 ISmartCard 인터페이스를 메인 개체로부터 가져오기만 하면 내부적으로 초기화가 이루어져 사용 가능한 상태가 된다.

■ **Example**

```
Dim objUCBioBSP As UCBioBSPCOMLib.UCBioBSP
Dim objSmartCard As ISmartCard          ' Declaration SmartCard object
…
' Create UCBioBSP object
Set objUCBioBSP = New UCBioBSPCOMLib.UCBioBSP
Set objSmartCard = objUCBioBSP.SmartCard
…
```

4.12.2. 스마트카드의 RF Power 켜기와 끄기

스마트카드를 사용하기 위해서는 우선 스마트카드 Reader기의 RF Power를 켜 주어야 한다. 이렇게 해 주어야지만 스마트카드로부터 데이터를 읽거나 쓰는 작업을 수행 할 수 있다. RF Power를 켜 주기 위해서는 ISmartCard의 RFPowerOn Method를 호출하면 된다.
스마트카드의 사용을 종료하려면 RFPowerOff Method를 호출하여 Reader기의 RF Power를 꺼 주면 된다.
Method 호출 전 LED라는 Property를 1로 지정 할 경우 함수의 성공과 실패 여부를 스마트카드 인식기의 LED에도 붉은색과 파란색으로 나타내 준다.

```
0 -  LED_TOGGLE
1 -  LED_NOT_TOGGLE
```

■ **Example - 1**

```
objSmartCard.LED = 0    ' NOT Toggle LED
objSmartCard.RFPowerOn
If objUCBioBSP.ErrorCode = 0 Then
    ' Succeeded to RFPowerOn
Else
    ' Failed to RFPowerOn
End If
```

■ **Example - 2**

```
objSmartCard.LED = 0    ' NOT Toggle LED
objSmartCard.RFPowerOff
If objUCBioBSP.ErrorCode = 0 Then
    ' Succeeded to RFPowerOff
Else
    ' Failed to RFPowerOff
End If
```

4.12.3. 스마트카드의 Serial 값 읽기

스마트카드에는 카드마다 고유한 Serial 값을 가지고 있다. 이 값을 읽기 위해서는 ReadSerial Method를 사용한다. 이 함수는 내부적으로 RFPowerOn 기능을 내장하고 있기 때문에 따로 RF Power를 켜주는 작업을 할 필요가 없으며 Key값을 알 필요도 없다. 간단하게 카드의 Serial 값을 읽기 위해 사용하면 편리하다. COM에서는 binary 형태의 Serial 값을 얻을 수도 있고 Value 형태의 Long 값으로 Serial 값을 얻을 수도 있다. 두 값은 동일하다.

■ **Example**

```
Dim serial() As Byte
Dim serialValue As Long

objSmartCard.ReadSerial

If objUCBioBSP.ErrorCode = 0 Then
    ' Succeeded to read serial
    serial = objSmartCard.ResultBuffer    ' binary type of serial
    serialValue = objSmartCard.serial     ' value type of serial
Else
    ' Failed to read serial
End If
```

4.12.4. Block 값 읽기 및 쓰기

스마트카드의 EEPROM에 적혀있는 각 Block의 값을 읽고 쓰기 위해서는 반드시 RF Power가 켜져 있어야 한다. 값을 읽기 위해서는 ReadBlock Method를 사용하고 값을 쓰기 위해서는 WriteBlock Method를 사용한다. Method를 호출 하기 전에 Block의 접근 권한과 그 권한에 맞는 Keyr값을 Property로 지정한 다음 호출하여야 한다. 각 Method의 인자로는 읽거나 쓰기를 원하는 Sector 번호와 Block 번호를 지정하면 된다.
사용 예는 다음과 같다.

■ **Example - 1**

```
Dim blockData() As Byte
Dim key(6) As Byte
' Set key value to key array
…
objSmartCard.AuthMode = &H60      ' 0x60 - Use Key A
objSmartCard.KeyA = key

nSectorNum = 0
nBlockNum = 0

objSmartCard.ReadBlock nSectorNum, nBlockNum

If objUCBioBSP.ErrorCode = 0 Then
    ' Succeeded to read block
   blockData = objSmartCard.ResultBuffer
Else
    ' Failed to read block
End If
```

■ **Example - 2**

```
Dim blockData(16) As Byte
Dim key(6) As Byte

' Set block data to blockData array
' Set key value to key array
…
objSmartCard.AuthMode = &H60      ' 0x60 - Use Key A
objSmartCard.KeyA = key

nSectorNum = 0
nBlockNum = 0
```

```
objSmartCard.WriteBlock nSectorNum, nBlockNum, blockData
If objUCBioBSP.ErrorCode = 0 Then
    ' Succeeded to write block
Else
    ' Failed to write block
End If
```

이 외의 다른 스마트카드 관련 함수들에 대해서는 API 레퍼런스를 참조하기 바란다.

5.API Reference for DLL

이 장에서는 DLL 모듈인 UCBioBSP.dll을 사용하기 위한 Type 및 API들에 대해 설명한다.

5.1. Type definitions

5.1.1. Basic types

UCBioAPI_Basic.h에 선언되어 있으며 기본 Type들에 대해 정의하고 있다. OS나 CPU 독립적인 개발을 위해 기본 Type들에 대해 재정의 하고 있다. 아래 설명은 일반적인 Windows 상에서 C++로 개발하는 것을 기준으로 설명한다.

■ **UCBioAPI_SINT8 / UCBioAPI_SINT16 / UCBioAPI_SINT32**
부호 있는 1byte / 2bytes / 4bytes 값

■ **UCBioAPI_UINT8 / UCBioAPI_UINT16 / UCBioAPI_UINT32 / UCBioAPI_UINT64**
부호 없는 1byte / 2bytes / 4bytes / 8bytes 값

■ **UCBioAPI_SINT / UCBioAPI_UINT**
OS에 따라 달라지는 int / unsigned int 값. 32bits OS의 경우 4bytes로 동작하며 64bits OS의 경우에는 8bytes로 동작한다.

■ **UCBioAPI_VOID_PTR**
void*를 의미

■ **UCBioAPI_BOOL**
UCBioAPI_FALSE(0) / UCBioAPI_TRUE(1) 값을 가질 수 있다. int와 동일하게 처리됨.

■ **UCBioAPI_CHAR / UCBioAPI_CHAR_PTR**
char와 char*를 의미. 1byte 문자 및 문자열 값.

■ **UCBioAPI_NULL**
NULL을 의미. ((void*)0)의 값으로 정의된다.

■ **UCBioAPI_HWND**
윈도우의 Handle을 의미하는 HWND 값.

5.1.2. General types

UCBioAPI_Type.h에 선언되어 있으며 일반적인 Type들에 대해 정의하고 있다.

■ **UCBioAPI_FIR_VERSION**

Prototype:

```
typedef UCBioAPI_UINT16  UCBioAPI_FIR_VERSION;
```

Description:

FIR 데이터의 버전값을 나타낸다.

■ **UCBioAPI_VERSION**

Prototype:

```
typedef struct ucbioapi_version {
   UCBioAPI_UINT16     Major1;
   UCBioAPI_UINT16     Major2;
   UCBioAPI_UINT16     Minor1;
   UCBioAPI_UINT16     Minor2;
} UCBioAPI_VERSION, *UCBioAPI_VERSION_PTR;
```

Description:

BSP의 버전값을 담는 4단계의 구조체. 만약 v3.4.0.2 이라면 Major1에 3, Major2에 4, Minor1에 0, Minor2에 2 라는 값이 담기게 된다.

■ **UCBioAPI_FIR_DATA_TYPE**

Prototype:

```
typedef UCBioAPI_UINT16   UCBioAPI_FIR_DATA_TYPE;
```

Description:

FIR의 데이터 Type을 나타낸다. 가질 수 있는 값으로는 다음과 같다. 각각의 값들은 OR 연산을 통해 중복 지정이 가능하다.

```
#define UCBioAPI_FIR_DATA_TYPE_RAW           (0x00)
#define UCBioAPI_FIR_DATA_TYPE_INTERMEDIATE  (0x01)
#define UCBioAPI_FIR_DATA_TYPE_PROCESSED     (0x02)
#define UCBioAPI_FIR_DATA_TYPE_ENCRYPTED     (0x10)
```

일반적으로 FIR이 만들어지면 (0x12)의 값을 가지게 된다.

■ **UCBioAPI_FIR_PURPOSE**

Prototype:

```
typedef UCBioAPI_UINT16   UCBioAPI_FIR_PURPOSE;
```

Description:

FIR의 데이터 Purpose를 나타낸다. 가질 수 있는 값으로는 다음과 같다. 이 값은 FIR 데이터의 참조용으로만 사용되고 인증에는 영향을 미치지 않는다.

```
#define UCBioAPI_FIR_PURPOSE_VERIFY                          (0x01)
#define UCBioAPI_FIR_PURPOSE_IDENTIFY                        (0x02)
#define UCBioAPI_FIR_PURPOSE_ENROLL                          (0x03)
#define UCBioAPI_FIR_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY    (0x04)
#define UCBioAPI_FIR_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY  (0x05)
#define UCBioAPI_FIR_PURPOSE_AUDIT                           (0x06)
#define UCBioAPI_FIR_PURPOSE_UPDATE                          (0x10)
```

■ **UCBioAPI_FIR_QUALITY**

Prototype:

```
typedef UCBioAPI_UINT16   UCBioAPI_FIR_QUALITY;
```

Description:

FIR의 데이터 품질값을 나타낸다. 가질 수 있는 값의 범위는 0~100이며 값이 클수록 좋은 품질의 지문 데이터임을 의미한다.

■ **UCBioAPI_FIR_PRIVILEGE**

Prototype:

```
typedef UCBioAPI_UINT16   UCBioAPI_FIR_PRIVILEGE;
```

Description:

FIR의 데이터 권한을 나타낸다. 가질 수 있는 값으로는 다음과 같다. 이 값은 참조용으로만 사용되고 인증에는 영향을 미치지 않는다.

```
#define UCBioAPI_FIR_PRIVILEGE_NOT_USED      (0)
#define UCBioAPI_FIR_PRIVILEGE_LOWEST        (1)
#define UCBioAPI_FIR_PRIVILEGE_LOWER         (2)
#define UCBioAPI_FIR_PRIVILEGE_LOW           (3)
#define UCBioAPI_FIR_PRIVILEGE_BELOW_NORMAL  (4)
#define UCBioAPI_FIR_PRIVILEGE_NORMAL        (5)
#define UCBioAPI_FIR_PRIVILEGE_ABOVE_NORMAL  (6)
```

```
#define UCBioAPI_FIR_PRIVILEGE_HIGH          (7)
#define UCBioAPI_FIR_PRIVILEGE_HIGHER        (8)
#define UCBioAPI_FIR_PRIVILEGE_HIGHEST       (9)
```

■ **UCBioAPI_FIR_DATE**

Prototype:

```
typedef struct ucbioapi_fir_date {
   UCBioAPI_UINT16     Year;    // 0 = 2000 / 1 = 2001 / ...
   UCBioAPI_UINT8      Month;
   UCBioAPI_UINT8      Day;
} UCBioAPI_FIR_DATE;
```

Description:

FIR의 데이터 날짜 정보를 담는 구조체이다. Year에는 2000년부터 시작되는 값이 들어가므로 2008년의 경우 8이라고만 하면 된다. UCBioAPI_FIR_OPTIONAL_DATA 구조체에 사용되어 UCBioAPI_VerifyMatchEx 함수 사용 시 인증용으로 사용되어 질 수 있다.

■ **UCBioAPI_FIR_UUID_INFO**

Prototype:

```
typedef struct ucbioapi_fir_ott_info {
   UCBioAPI_UINT32     Index;
   UCBioAPI_UINT8      UUID[16];
} UCBioAPI_FIR_UUID_INFO, *UCBioAPI_FIR_UUID_INFO_PTR;
```

Description:

FIR의 고유한 UUID(Universally Unique Identifier) 정보를 담는 구조체이다.
일반적으로는 사용되지 않지만 UCBioAPI_FIR_OPTIONAL_DATA 구조체에 사용되어 UCBioAPI_VerifyMatchEx 함수 사용 시 인증용으로 사용되어 질 수 있다.

■ **UCBioAPI_OPTIONAL_DATA_TYPE**

Prototype:

```
typedef UCBioAPI_UINT32  UCBioAPI_OPTIONAL_DATA_TYPE;
```

Description:

UCBioAPI_VerifyMatchEx 함수를 이용해 인증을 수행 할 때 Optional Data 중 어떤 것을 추가 인증 데이터로 사용 할지를 결정하는 Type이다. UCBioAPI_FIR_OPTIONAL_DATA 구조체의 각 필드들을 의미한다. 각각의 값들은 OR 연산을 통해 중복 지정이 가능하다. 가질 수 있는 값으로는 다음과 같다.

```
#define UCBioAPI_OPTIONAL_DATA_TYPE_UUID            (0x00000001)
#define UCBioAPI_OPTIONAL_DATA_TYPE_PIN1            (0x00000002)
#define UCBioAPI_OPTIONAL_DATA_TYPE_PIN2            (0x00000004)
#define UCBioAPI_OPTIONAL_DATA_TYPE_PRIVILEGE       (0x00000008)
#define UCBioAPI_OPTIONAL_DATA_TYPE_SITEID          (0x00000010)
#define UCBioAPI_OPTIONAL_DATA_TYPE_ISSUEDATE       (0x00000020)
#define UCBioAPI_OPTIONAL_DATA_TYPE_EXPIREDATE      (0x00000040)
#define UCBioAPI_OPTIONAL_DATA_TYPE_ALL             (0xffffffff)
```

■ **UCBioAPI_FIR_OPTIONAL_DATA**

Prototype:

```
typedef struct ucbioapi_fir_optional_data {
   UCBioAPI_UINT32              Length;
   UCBioAPI_FIR_UUID_INFO       UUIDInfo;
   UCBioAPI_UINT32              PIN1;
   UCBioAPI_UINT32              PIN2;
   UCBioAPI_UINT32              Privilege;
   UCBioAPI_UINT32              SiteID;
   UCBioAPI_FIR_DATE            IssueDate;
   UCBioAPI_FIR_DATE            ExpireDate;
   UCBioAPI_UINT32              Reserved;
} UCBioAPI_FIR_OPTIONAL_DATA, *UCBioAPI_FIR_OPTIONAL_DATA_PTR;
```

Description:

UCBioAPI_VerifyMatchEx 함수를 이용해 인증을 수행 할 때 추가 인증 데이터로 사용할 Optional Data에 대한 구조체이다. 각 필드들에 대한 설명은 다음과 같다.

Length:

구조체의 길이값으로 sizeof(UCBioAPI_FIR_OPTIONAL_DATA)의 값을 가진다.

UUIDInfo:

FIR의 UUID 값을 가지는 구조체. UCBioAPI_FIR_UUID_INFO 설명 참조.

PIN1, PIN2:

Personal Identification Number를 나눠 담을 수 있는 값.

Privilege:

FIR의 권한 값. 인증 시 권한레벨이 낮을 경우 인증이 되지 않도록 할 수 있다.

SiteID:

FIR 데이터가 사용되어야 할 Site의 ID를 지정하여 특정 Site에서만 사용되게 할 수 있다.

IssueDate, ExporeDate:

FIR 데이터가 만들어진 날짜와 사용가능 한 날짜를 지정 할 수 있다.

Reserved:

예약 공간

■ **UCBioAPI_FIR_HEADER**

Prototype:

```
typedef struct ucbioapi_fir_header {
   UCBioAPI_UINT32              Length;
   UCBioAPI_UINT32              DataLength;
   UCBioAPI_FIR_VERSION         Version;
   UCBioAPI_FIR_DATA_TYPE       DataType;
   UCBioAPI_FIR_PURPOSE         Purpose;
   UCBioAPI_FIR_QUALITY         Quality;
   UCBioAPI_FIR_OPTIONAL_DATA   OptionalData;
   UCBioAPI_UINT32              Reserved;
} UCBioAPI_FIR_HEADER, *UCBioAPI_FIR_HEADER_PTR;
```

Description:

FIR 데이터의 Header 정보를 담는 구조체이다. UCBioAPI_FIR 구조체에 사용된다. 각 필드들에 대한 설명은 다음과 같다.

Length:

구조체의 길이값으로 sizeof(UCBioAPI_FIR_HEADER)의 값을 가진다.

DataLength:

실제 FIR 데이터가 담기는 길이 값을 가진다.

Version:

FIR 데이터의 버전 정보를 가진다. UCBioAPI_FIR_VERSION 참조.

DataType:

FIR 데이터의 Type 값을 가진다. UCBioAPI_FIR_DATA_TYPE 참조.

Purpose:

FIR 데이터의 Purpose 값을 가진다. UCBioAPI_FIR_PURPOSE 참조.

Quality:

FIR 데이터의 품질 값을 가진다. UCBioAPI_FIR_QUALITY 참조.

OptionalData:

FIR 데이터의 추가 인증 정보 값을 가지는 구조체. UCBioAPI_FIR_OPTIONAL_DATA 참조.

Reserved:

예약 공간

■ **UCBioAPI_FIR_DATA**

Prototype:

```
typedef UCBioAPI_UINT8   UCBioAPI_FIR_DATA;
```

Description:
FIR의 실제 데이터 블록의 Type을 나타낸다.

■ **UCBioAPI_FIR_FORMAT**

Prototype:

```
typedef UCBioAPI_UINT32  UCBioAPI_FIR_FORMAT;
```

Description:
FIR의 데이터 포맷을 나타낸다. 가질 수 있는 값으로는 다음과 같다. 향후 FIR 데이터의 구조가 바뀔 경우 이 포맷값이 바뀌어 하위 호환성을 유지 할 수 있다. 현재는 STANDARD 포맷만 지원하고 있다.

```
#define UCBioAPI_FIR_FORMAT_STANDARD     (1)
```

■ **UCBioAPI_FIR**

Prototype:

```
typedef struct ucbioapi_fir {
   UCBioAPI_FIR_FORMAT    Format;
   UCBioAPI_FIR_HEADER    Header;
   UCBioAPI_FIR_DATA*     Data;
} UCBioAPI_FIR, *UCBioAPI_FIR_PTR;
```

Description:
FIR 데이터를 나타내는 구조체. 실제 FIR의 Binary 데이터를 담게 된다.

Format:
FIR의 데이터 포멧. UCBioAPI_FIR_FORMAT 참조.
Header:
FIR의 각종 정보를 담고 있는 Header. UCBioAPI_FIR_HEADER 참조.
Data:
FIR의 실제 데이터를 담고 있는 Binary 블록. 이 데이터의 길이 값은 Header의 DataLength에 기록된다.

■ **UCBioAPI_FIR_PAYLOAD**

Prototype:

```
typedef struct ucbioapi_fir_payload {
   UCBioAPI_UINT32        Length;
```

```
    UCBioAPI_UINT8*        Data;
} UCBioAPI_FIR_PAYLOAD, *UCBioAPI_FIR_PAYLOAD_PTR;
```

Description:

Payload 정보를 담는 구조체. 지문 등록 시 FIR 내부에 사용자의 특정 값을 암호화 해 넣을 수 있고 나중에 사용자 인증이 성공하면 그 값을 꺼내 볼 수 있는데 그 정보를 저장하는 구조체이다. Payload 데이터는 어떠한 데이터도 사용 될 수 있기 때문에 사용자 인증 시 특정 고정된 값을 얻고자 할 경우 사용 할 수 있다.

Length

Payload 데이터의 길이 값.

Data:

실제 Payload 데이터 블록

■ UCBioAPI_HANDLE / UCBioAPI_HANDLE_PTR

Prototype:

```
typedef UCBioAPI_UINT  UCBioAPI_HANDLE;
typedef UCBioAPI_UINT* UCBioAPI_HANDLE_PTR;
```

Description:

UCBioAPI에서 사용되는 각종 Handle값을 정의.

Handle 값이 없을 경우에는 UCBioAPI_INVALID_HANDLE(0) 값이 사용된다.

```
#define UCBioAPI_INVALID_HANDLE     (0)
```

■ UCBioAPI_FIR_HANDLE / UCBioAPI_FIR_HANDLE_PTR

Prototype:

```
typedef UCBioAPI_UINT  UCBioAPI_FIR_HANDLE;
typedef UCBioAPI_UINT* UCBioAPI_FIR_HANDLE_PTR;
```

Description:

FIR의 데이터의 Handle 값을 정의.

■ UCBioAPI_FIR_SECURITY_LEVEL

Prototype:

```
typedef UCBioAPI_UINT32   UCBioAPI_FIR_SECURITY_LEVEL;
```

Description:

FIR 데이터의 인증 보안 레벨을 지정한다. 가질 수 있는 값은 다음과 같다.

```
#define UCBioAPI_FIR_SECURITY_LEVEL_LOWEST          (1)
#define UCBioAPI_FIR_SECURITY_LEVEL_LOWER           (2)
#define UCBioAPI_FIR_SECURITY_LEVEL_LOW             (3)
#define UCBioAPI_FIR_SECURITY_LEVEL_BELOW_NORMAL    (4)
#define UCBioAPI_FIR_SECURITY_LEVEL_NORMAL          (5)
#define UCBioAPI_FIR_SECURITY_LEVEL_ABOVE_NORMAL    (6)
#define UCBioAPI_FIR_SECURITY_LEVEL_HIGH            (7)
#define UCBioAPI_FIR_SECURITY_LEVEL_HIGHER          (8)
#define UCBioAPI_FIR_SECURITY_LEVEL_HIGHEST         (9)
```

■ UCBioAPI_TEMPLATE_FORMAT

Prototype:

```
typedef UCBioAPI_UINT32   UCBioAPI_TEMPLATE_FORMAT;
```

Description:

FIR 데이터의 Template 포맷을 지정한다. 가질 수 있는 값은 다음과 같다.

```
#define UCBioAPI_TEMPLATE_FORMAT_UNION400           (0)
#define UCBioAPI_TEMPLATE_FORMAT_ISO500             (1)
#define UCBioAPI_TEMPLATE_FORMAT_ISO600             (2)
```

■ UCBioAPI_INIT_INFO_0

Prototype:

```
typedef struct ucbioapi_init_info_0 {
   UCBioAPI_UINT32              StructureType;
   UCBioAPI_UINT32              MaxFingersForEnroll;
   UCBioAPI_UINT32              NecessaryEnrollNum;
   UCBioAPI_UINT32              SamplesPerFinger;
   UCBioAPI_UINT32              DefaultTimeout;
   UCBioAPI_FIR_SECURITY_LEVEL  SecurityLevelForEnroll;
   UCBioAPI_FIR_SECURITY_LEVEL  SecurityLevelForVerify;
   UCBioAPI_FIR_SECURITY_LEVEL  SecurityLevelForIdentify;
   UCBioAPI_TEMPLATE_FORMAT     TemplateFormat;
   UCBioAPI_UINT32              Reserved1;
   UCBioAPI_UINT32              Reserved2;
} UCBioAPI_INIT_INFO_0, *UCBioAPI_INIT_INFO_PTR_0;
```

Description:

UCBioAPI SDK의 기본 초기 설정 값을 담는 구조체. 각각의 값들에 대한 설명은 다음과 같다. UCBioAPI_GetInitInfo / UCBioAPI_SetInitInfo 함수에서 사용된다.

StructureType:

구조체 Type. 향후 추가되는 초기 설정 값이 있을 경우 구조체 1, 2, .. 등과 같이 증가될 수 있다. 현재는 0만 사용된다.

MaxFingersForEnroll:

등록 시 최대 등록 가능한 손가락 개수를 지정. 여기에서 지정된 손가락 개수가 등록되면 더 이상의 손가락을 등록하지 못하게 된다. 기본 값은 10이다.

NecessaryEnrollNum:

등록 시 반드시 등록해야 하는 최소 손가락의 개수를 지정. 만약 2라고 설정하면 최소 2개 이상의 손가락을 등록해야만 지문 등록을 종료 할 수 있다. 때문에 이 값은 MaxFingersForEnroll 값보다는 반드시 같거나 작은 값이어야 한다. 기본 값은 1이다.

SamplesPerFinger:

손가락당 지문 Sample의 개수를 지정. 현재는 2로 고정되어 있으며 변경 불가함.

DefaultTimeout:

지문 등록 및 인증 시 지문 입력 대기시간을 위한 기본 Timeout 값을 지정. 단위는 millisecond로 1초의 경우 1,000이라고 지정하면 됨. 기본 값은 10,000(10초)이다.

SecurityLevelForEnroll / SecurityLevelForVerify / SecurityLevelForIdentify:

지문 등록, 인증 시 사용되는 기본 보안 레벨을 지정. UCBioAPI_FIR_SECURITY_LEVEL 참조. 기본값은 순서대로 5 / 5 / 6 이 사용된다.

TemplateFormat:

FIR 데이터의 Template 포맷을 지정.

기본값은 `UCBioAPI_TEMPLATE_FORMAT_UNION400` 이다.

Reserved1 / Reserved2:

예약된 값.

■ UCBioAPI_DEVICE_ID

Prototype:

```
typedef UCBioAPI_UINT16   UCBioAPI_DEVICE_ID;
```

Description:

디바이스의 ID 값을 정의. 이 값은 2bytes로 되어 있는데 하위 1byte가 각 디바이스의 종류를 나타내는 UCBioAPI_DEVICE_NAME을 의미하고 상위 1byte는 각 디바이스별 Instance 값을 나타낸다. 가질 수 있는 값으로는 다음과 같다. 현재 시스템에 있는 디바이스 중 가장 최근에 사용된 디바이스를 자동으로 사용하고자 할 경우에는 AUTO(0x00ff)를 지정하면 된다.

```
#define UCBioAPI_DEVICE_ID_NONE           (0x0000)
```

```
#define UCBioAPI_DEVICE_ID_AUTO          (0x00ff)
```

■ **UCBioAPI_DEVICE_NAME**

Prototype:

```
typedef UCBioAPI_UINT8    UCBioAPI_DEVICE_NAME;
```

Description:

UCBioAPI_DEVICE_ID의 하위 byte로 사용되는 값으로 각 디바이스의 종류를 나타낸다. 가질 수 있는 값으로는 다음과 같다. 향후 디바이스가 추가 되면 이 값은 더 늘어나게 된다.

```
#define UCBioAPI_DEVICE_NAME_FOH01           (0x01)
#define UCBioAPI_DEVICE_NAME_FOM01           (0x02)
#define UCBioAPI_DEVICE_NAME_FOH03           (0x03)
#define UCBioAPI_DEVICE_NAME_HAM500          (0x04)
#define UCBioAPI_DEVICE_NAME_FOH01A          (0x05)
#define UCBioAPI_DEVICE_NAME_FOM01A          (0x06)
#define UCBioAPI_DEVICE_NAME_FPR02           (0x07)
#define UCBioAPI_DEVICE_NAME_FSH01RF         (0x08)
#define UCBioAPI_DEVICE_NAME_FOH01RF         (0x09)
#define UCBioAPI_DEVICE_NAME_FR100           (0x0a) // 10
#define UCBioAPI_DEVICE_NAME_FPR02LFD        (0x0b) // 11
#define UCBioAPI_DEVICE_NAME_FOH01RFL        (0x0c) // 12
#define UCBioAPI_DEVICE_NAME_FSH01SC         (0x0d) // 13
#define UCBioAPI_DEVICE_NAME_FPR02_V30       (0x0e) // 14
```

■ **UCBioAPI_DEVICE_INFO_0**

Prototype:

```
typedef struct ucbioapi_device_info_0 {
   UCBioAPI_UINT32     StructureType;
   UCBioAPI_UINT32     ImageWidth;
   UCBioAPI_UINT32     ImageHeight;
   UCBioAPI_UINT32     Brightness;
   UCBioAPI_UINT32     Contrast;
   UCBioAPI_UINT32     Gain;
} UCBioAPI_DEVICE_INFO_0, *UCBioAPI_DEVICE_INFO_PTR_0;
```

Description:

디바이스 정보에 대한 값을 담는 구조체.

UCBioAPI_GetDeviceInfo / UCBioAPI_SetDeviceInfo 함수에서 사용된다. 각각의 값들에 대한

설명은 다음과 같다.

StructureType:
구조체 Type. 향후 추가되는 초기 설정 값이 있을 경우 구조체 1, 2, .. 등과 같이 증가될 수 있다. 현재는 0만 사용된다.
ImageWidth / ImageHeight:
디바이스로부터 얻을 수 있는 이미지의 크기 값. 디바이스마다 다른 값을 가질 수 있다. 이 값은 읽기 전용으로 값을 변경 할 수는 없다.
Brightness / Contrast / Gain:
디바이스의 밝기 / 대비 / Gain 값을 얻거나 지정 할 수 있다. 현재는 Gain에 대한 값만 유효하게 사용되며 디바이스에 따라 사용 될 수 있는 값이 다르다.

■ **UCBioAPI_DEVICE_INFO_EX**

Prototype:

```
typedef struct ucbioapi_deviceinfoex {
  UCBioAPI_DEVICE_ID     NameID;
  UCBioAPI_UINT16        Instance;

  UCBioAPI_CHAR          Name[64];
  UCBioAPI_CHAR          Description[256];
  UCBioAPI_CHAR          Dll[64];
  UCBioAPI_CHAR          Sys[64];

  UCBioAPI_UINT32        Brightness;
  UCBioAPI_UINT32        Contrast;
  UCBioAPI_UINT32        Gain;

  UCBioAPI_UINT32        Reserved[8];
} UCBioAPI_DEVICE_INFO_EX, *UCBioAPI_DEVICE_INFO_EX_PTR;
```

Description:
디바이스 정보에 대한 좀 더 자세한 값을 담는 구조체.
UCBioAPI_EnumerateDevice 함수에서 디바이스 목록을 얻어올 때 이 구조체에 값이 담겨져 오게 된다. 각각의 값들에 대한 설명은 다음과 같다.

NameID:
디바이스의 종류를 지정. UCBioAPI_DEVICE_ID 참조.
Instance:
디바이스별 Instacne 값을 지정. UCBioAPI_DEVICE_ID 참조.
Name:

디바이스의 이름이 String 형태로 들어있다.

Description:

디바이스에 대한 설명이 String 형태로 들어있다.

Dll / Sys:

디바이스가 사용하는 Dll, Sys 파일에 대한 정보가 들어있다.

Brightness / Contrast / Gain:

디바이스의 밝기 / 대비 / Gain 값을 얻거나 지정 할 수 있다. 현재는 Gain에 대한 값만 유효하게 사용되며 디바이스에 따라 사용 될 수 있는 값이 다르다.

Reserved:

예약 된 값.

■ **UCBioAPI_RETURN**

Prototype:

```
typedef UCBioAPI_UINT32  UCBioAPI_RETURN;
```

Description:

UCBioAPI SDK의 함수들이 리턴하는 값을 정의. 일반적으로 UCBioAPI SDK의 오류값을 담게 된다. 자세한 오류 값들은 ERROR 정의를 참조.

■ **UCBioAPI_FIR_TEXTENCODE**

Prototype:

```
typedef struct ucbioapi_fir_textencode {
   UCBioAPI_BOOL          IsWideChar;
   UCBioAPI_CHAR_PTR      TextFIR;
} UCBioAPI_FIR_TEXTENCODE, *UCBioAPI_FIR_TEXTENCODE_PTR;
```

Description:

FIR 데이터를 Text 형태로 나타낸 값을 저장하는 구조체. FIR 데이터는 Binary 형태와 Text 형태, 이렇게 두 가지 형태로 나타낼 수 있으며 어떻게 사용하던 동일하다. Binary 형태의 데이터를 사용하기 곤란할 경우 편리하게 Text String 형태로 FIR을 저장하고자 할 경우 사용하면 된다. UCBioAPI_GetTextFIRFromHandle 등과 같은 함수를 이용해 얻어 질 수 있다.

IsWideChar:

TextFIR이 유니코드인 2bytes 문자열로 되어있는지를 나타낸다.

TextFIR:

실제 FIR 데이터가 Text String 형태로 들어가 있다.

■ **UCBioAPI_INPUT_FIR_FORM**

Prototype:

```
typedef UCBioAPI_UINT8   UCBioAPI_INPUT_FIR_FORM;
```

Description:

FIR 데이터를 함수 인자로 넘길 경우 사용되는 UCBioAPI_INPUT_FIR 구조체의 Type을 정의. FIR 데이터는 FIR Handle, Binary FIR, Text FIR과 같이 총 3개의 각기 다른 형태로 사용될 수 있는데 그것의 종류를 지정 할 수 있다. 사용 될 수 있는 값은 다음과 같다.

```
#define UCBioAPI_FIR_FORM_HANDLE       (0x02)
#define UCBioAPI_FIR_FORM_FULLFIR      (0x03)
#define UCBioAPI_FIR_FORM_TEXTENCODE   (0x04)
```

■ UCBioAPI_INPUT_FIR

Prototype:

```
typedef struct ucbioapi_input_fir {
   UCBioAPI_INPUT_FIR_FORM  Form;
   union {
      UCBioAPI_FIR_HANDLE_PTR          FIRinBSP;
      UCBioAPI_VOID_PTR                FIR;
      UCBioAPI_FIR_TEXTENCODE_PTR      TextFIR;
   } InputFIR;
} UCBioAPI_INPUT_FIR, *UCBioAPI_INPUT_FIR_PTR;
```

Description:

FIR 데이터를 함수 인자로 넘길 경우 사용되는 구조체. FIR 데이터는 FIR Handle, Binary FIR, Text FIR과 같이 총 3개의 각기 다른 형태로 사용될 수 있는데 그것을 하나의 입력 값으로 동시에 표현하기 위해 이 구조체가 사용된다.

Form:

이 구조체가 가지는 FIR 데이터의 Type을 지정. UCBioAPI_INPUT_FIR_FORM 참조.

InputFIR:

실제 FIR 데이터를 지정하는 union 구조체. FIR Handle, Binary FIR, Text FIR의 값을 하나의 동일 주소 포인터로 저장해 사용 할 수 있게 되어 있다.

■ UCBioAPI_WINDOW_CALLBACK_PARAM_0

Prototype:

```
typedef struct ucbioapi_window_callback_param_0 {
   UCBioAPI_UINT32        dwQuality;
   UCBioAPI_UINT8*        lpImageBuf;
```

```
  UCBioAPI_UINT32       dwDeviceError;

  UCBioAPI_UINT32       dwReserved[8];
  UCBioAPI_VOID_PTR     lpReserved;
} UCBioAPI_WINDOW_CALLBACK_PARAM_0,
*UCBioAPI_WINDOW_CALLBACK_PARAM_PTR_0;
```

Description:
지문 획득 및 인증 시 불려지는 Callback 함수의 첫 번째 인자로 넘어오는 구조체.

dwQuality:
현재 이미지의 지문 품질 값을 가진다.
lpImageBuf:
현재 획득한 지문 이미지의 버퍼 포인터를 가진다.
dwDeviceError:
현재 디바이스로부터 발생한 오류 값이 있을 경우 그 값을 가진다.
dwReserved / lpReserved:
예약 된 값

■ **UCBioAPI_WINDOW_CALLBACK_PARAM_1**

Prototype:

```
typedef struct ucbioapi_window_callback_param_1 {
  UCBioAPI_UINT32       dwResult;

  UCBioAPI_UINT32       dwStartTime;
  UCBioAPI_UINT32       dwCapTime;
  UCBioAPI_UINT32       dwEndTime;

  UCBioAPI_UINT32       Reserved[8];
  UCBioAPI_VOID_PTR     lpReserved;
} UCBioAPI_WINDOW_CALLBACK_PARAM_1,
*UCBioAPI_WINDOW_CALLBACK_PARAM_PTR_1;
```

Description:
지문 등록 및 함수 종료 시 불려지는 Callback 함수의 첫 번째 인자로 넘어오는 구조체.

dwResult:
함수 종료 시 실제 그 함수가 최종 Reture 할 값을 담고 있다. 또한 지문 등록 시 지문 등록 과정에 따라 발생하는 Event 값을 가질 수 있다. 각각의 값은 ERROR 정의에 정의된 값들이 담겨질 수 있다.

dwStartTime / dwCapTime / dwEndTime:
함수 종료 시 각종 시간 값을 알 수 있다. 함수가 불려진 시간 / 실제 지문 획득이 시작된 시간 / 획득이 종료된 시간이 순서대로 들어가 있다.
Reserved / lpReserved:
예약 된 값.

■ **UCBioAPI_WINDOW_CALLBACK_0 / UCBioAPI_WINDOW_CALLBACK_1**

Prototype:

```
typedef UCBioAPI_RETURN (WINAPI* UCBioAPI_WINDOW_CALLBACK_0)
(UCBioAPI_WINDOW_CALLBACK_PARAM_PTR_0, UCBioAPI_VOID_PTR);

typedef UCBioAPI_RETURN (WINAPI* UCBioAPI_WINDOW_CALLBACK_1)
(UCBioAPI_WINDOW_CALLBACK_PARAM_PTR_1, UCBioAPI_VOID_PTR);
```

Description:
지문 획득 및 등록 시 불려지는 Callback 함수에 대한 함수 포인터 정의.
각각의 함수는 UCBioAPI_WINDOW_OPTION의 CaptureCallBackInfo와 FinishCallBackInfo에 지정 할 수 있다. 첫 번째 인자로는 각 Callback 함수에 맞는 정보가 넘어오고 두 번째 인자로는 사용자가 넘겨준 포인터의 값이 넘어온다.
Callback 함수는 0을 리턴하게 되는데 만약 0 이외의 값을 리턴하게 되면 현재 진행중인 함수를 종료시키게 된다.

■ **UCBioAPI_CALLBACK_INFO_0 / UCBioAPI_CALLBACK_INFO_1**

Prototype:

```
typedef struct ucbioapi_callback_info_0 {
  UCBioAPI_UINT32              CallBackType;
  UCBioAPI_WINDOW_CALLBACK_0   CallBackFunction;
  UCBioAPI_VOID_PTR            UserCallBackParam;
} UCBioAPI_CALLBACK_INFO_0, *UCBioAPI_CALLBACK_INFO_PTR_0;

typedef struct ucbioapi_callback_info_1 {
  UCBioAPI_UINT32              CallBackType;
  UCBioAPI_WINDOW_CALLBACK_1   CallBackFunction;
  UCBioAPI_VOID_PTR            UserCallBackParam;
} UCBioAPI_CALLBACK_INFO_1, *UCBioAPI_CALLBACK_INFO_PTR_1;
```

Description:
지문 획득 시 불려질 Callback 함수에 대한 정보를 지정하기 위한 구조체.

CallBackType:
Callback 함수의 Type을 지정. 이 값은 UCBioAPI_CALLBACK_INFO_0에서는 항상 0이고 UCBioAPI_CALLBACK_INFO_1에서는 항상 1의 값을 가진다.
CallBackFunction:
실제 Callback으로 불려질 함수를 지정.
UserCallBackParam:
Callback 함수의 두 번째 인자로 넘겨질 사용자 인자를 지정.

■ **UCBioAPI_WINDOW_STYLE**

Prototype:

```
typedef UCBioAPI_UINT32  UCBioAPI_WINDOW_STYLE;
```

Description:
지문 등록, 인증 시 사용되는 User Interface의 윈도우 스타일을 지정한다.
이 값은 UCBioAPI_WINDOW_OPTION 구조체에 사용된다. 사용 될 수 있는 값은 다음과 같다.

```
#define UCBioAPI_WINDOW_STYLE_POPUP          (0)
#define UCBioAPI_WINDOW_STYLE_INVISIBLE      (1)

#define UCBioAPI_WINDOW_STYLE_NO_FPIMG       (0x00010000)
#define UCBioAPI_WINDOW_STYLE_NO_WELCOME     (0x00020000)
#define UCBioAPI_WINDOW_STYLE_NO_TOPMOST     (0x00040000)
```

UCBioAPI_WINDOW_STYLE_POPUP:
UCBioAPI_WINDOW_STYLE_INVISIBLE:
POPUP으로 지정 할 경우 일반적으로 새로운 윈도우가 떠서 지문 등록 및 인증을 수행한다. 하지만 UCBioAPI_Capture 함수와 UCBioAPI_Verify 함수의 경우 INVISIBLE 값을 지정할 수 있는데 그렇게 할 경우 화면에 UI를 표시 하지 않으면서 지문 입력이나 인증을 수행 할 수 있다. 또한 이렇게 INVISIBLE로 지정 할 경우 UCBioAPI_WINDOW_OPTION 구조체의 값 중 FingerWnd의 값이 NULL이 아니라면 그 윈도우에 지문 이미지를 표시하도록 할 수도 있다.

아래 3개의 Style은 OR 연산자를 이용해 중복 지정이 가능한 값들이다. 각각의 값들은 다음과 같은 의미이다.

UCBioAPI_WINDOW_STYLE_NO_FPIMG:
보안상 지문 이미지를 화면에 표시하고자 하지 않을 경우 이 Style을 지정한다.
UCBioAPI_WINDOW_STYLE_NO_WELCOME:
지문 등록 UI에서 첫 번째 Page인 Welcome Page를 표시하지 않는다.

UCBioAPI_WINDOW_STYLE_NO_TOPMOST:
현재는 UI가 모두 최상위 윈도우로 뜨게 되어있으나 이것을 일반 윈도우로 뜨게 할 경우 이 Style을 지정한다.

■ **UCBioAPI_WINDOW_OPTION**

Prototype:

```
typedef struct ucbioapi_window_option {
   UCBioAPI_UINT32               Length;
   UCBioAPI_WINDOW_STYLE         WindowStyle;
   UCBioAPI_HWND                 ParentWnd;
   UCBioAPI_HWND                 FingerWnd;
   UCBioAPI_CALLBACK_INFO_0      CaptureCallBackInfo;
   UCBioAPI_CALLBACK_INFO_1      FinishCallBackInfo;
   UCBioAPI_CHAR_PTR             CaptionMsg;
   UCBioAPI_CHAR_PTR             CancelMsg;
   UCBioAPI_WINDOW_OPTION_PTR_2 Option2;
} UCBioAPI_WINDOW_OPTION, *UCBioAPI_WINDOW_OPTION_PTR;
```

Description:
지문 등록, 인증 시 사용되는 User Interface의 각종 정보들을 지정하는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

Length:
구조체의 길이 값. sizeof(UCBioAPI_WINDOW_OPTION)의 값을 같는다.
WindowStyle:
UI의 윈도우 Style의 값을 지정. UCBioAPI_WINDOW_STYLE 참조.
기본값은 UCBioAPI_WINDOW_STYLE_POPUP(0) 값을 가진다.
ParentWnd:
지문 등록 및 인증 윈도우가 뜰 때 그 기반이 되는 부모 윈도우의 Handle을 지정.
기본값은 NULL 값을 가진다.
FingerWnd:
WindowStyle이 UCBioAPI_WINDOW_STYLE_INVISIBLE(1) 값일 경우 지문 이미지를 그려줄 특정 윈도우의 Handle을 지정. 기본값은 NULL 값을 가진다.
CaptureCallBackInfo:
지문 획득 시마다 불려질 Callback 함수를 지정. UCBioAPI_CALLBACK_INFO_0 참조.
FinishCallBackInfo:
지문 등록 및 함수 종료 시 불려질 Callback 함수를 지정. UCBioAPI_CALLBACK_INFO_1 참조.
CaptionMsg:
지문 등록 시 사용자가 취소를 선택 할 경우 취소 메시지를 보여줄 윈도우의 Caption에

표시할 메시지를 지정. 기본값은 NULL이다.

CancelMsg:

지문 등록 시 사용자가 취소를 선택 할 경우 보여줄 취소 메시지를 지정. 기본값은 NULL 이다.

Option2:

추가적인 윈도우 옵션 값을 지정. 자세한 값은 UCBioAPI_WINDOW_OPTION_2 참조. 기본값은 NULL이다.

■ **UCBioAPI_WINDOW_OPTION_2**

Prototype:

```
typedef struct ucbioapi_window_option_2 {
  UCBioAPI_UINT8         FPForeColor[3];
  UCBioAPI_UINT8         FPBackColor[3];
  UCBioAPI_UINT8         DisableFingerForEnroll[10];
  UCBioAPI_UINT32        Reserved1[4];
  UCBioAPI_VOID_PTR      Reserved2;
} UCBioAPI_WINDOW_OPTION_2, *UCBioAPI_WINDOW_OPTION_PTR_2;
```

Description:

추가적인 윈도우 옵션 값을 지정하는 구조체. 각각의 값들에 대한 설명은 다음과 같다.

FPForeColor:

화면에 표시할 지문 이미지의 지문 색상 값을 RGB 값으로 지정.

FPBackColor:

화면에 표시할 지문 이미지의 지문 배경 값을 RGB 값으로 지정.

DisableFingerForEnroll

지문 등록 시 등록하지 못하게 할 손가락을 지정 할 수 있다. 총 10개의 값을 담을 수 있는 배열 값으로 각 손가락 값에 따른 등록 유무를 0 또는 1로 지정하며 된다.
1번 Index부터 오른손 엄지, 검지, 중지, 가락지, 약지 순으로 증가하며 6번 Index부터 왼손 엄지, 검지, 중지, 가락지, 약지 순으로 증가한다. 0번 Index는 사용하지 않는다.
예를 들어 왼손 엄지를 등록하지 못하게 하려면 다음과 같이 지정하면 된다.

```
DisableFingerForEnroll[UCBioAPI_FINGER_ID_LEFT_THUMB] = 1;
```

단, 지문 수정 모드일 경우 사용 금지를 한 손가락이라도 이미 등록된 손가락이라면 등록 상태를 표시해 준다.

Reserved1 / Reserved2

예약 된 값.

■ **UCBioAPI_TEMPLATE_TYPE**

Prototype:

```
typedef UCBioAPI_UINT32     UCBioAPI_TEMPLATE_TYPE;
```

Description:

FIR데이터를 Template 데이터로 변환 시 가질 수 있는 Template의 종류 값을 지정. 가질 수 있는 값은 다음과 같다. 추가적인 Type이 생길 경우 이 값이 추가 될 수 있다.

```
#define UCBioAPI_TEMPLATE_TYPE_SIZE400       (400)
#define UCBioAPI_TEMPLATE_TYPE_SIZE500       (500)
#define UCBioAPI_TEMPLATE_TYPE_SIZE600       (600)
#define UCBioAPI_TEMPLATE_TYPE_SIZE800       (800)
#define UCBioAPI_TEMPLATE_TYPE_SIZE320       (320)
#define UCBioAPI_TEMPLATE_TYPE_SIZE256       (256)
#define UCBioAPI_TEMPLATE_TYPE_FMR           (1)
#define UCBioAPI_TEMPLATE_TYPE_ANSI                (2)
```

SIZE400 / SIZE800 / SIZE320 / SIZE256은 각각의 크기를 가지는 Template을 의미한다. 스마트카드와 같이 한정된 저장공간에 지문을 저장해야 할 경우 SIZE256과 같은 데이터를 사용 할 수 있다. (단, 크기가 작아질수록 인증율도 같이 낮아진다고 볼 수 있다.)
FMR/ANSI Type은 지문 데이터에 대한 표준 데이터 포맷이다
SIZE500/SIZE600 Type은 지문 데이터에 대한 ISO 표준 데이터 포맷이다

■ **UCBioAPI_FINGER_ID**

Prototype:

```
typedef UCBioAPI_UINT8      UCBioAPI_FINGER_ID;
```

Description:

손가락의 ID 값을 지정한다. 가질 수 있는 값은 다음과 같다.

```
#define UCBioAPI_FINGER_ID_UNKNOWN           (0)
#define UCBioAPI_FINGER_ID_RIGHT_THUMB       (1)
#define UCBioAPI_FINGER_ID_RIGHT_INDEX       (2)
#define UCBioAPI_FINGER_ID_RIGHT_MIDDLE      (3)
#define UCBioAPI_FINGER_ID_RIGHT_RING        (4)
#define UCBioAPI_FINGER_ID_RIGHT_LITTLE      (5)
#define UCBioAPI_FINGER_ID_LEFT_THUMB        (6)
#define UCBioAPI_FINGER_ID_LEFT_INDEX        (7)
```

```
#define UCBioAPI_FINGER_ID_LEFT_MIDDLE       (8)
#define UCBioAPI_FINGER_ID_LEFT_RING         (9)
#define UCBioAPI_FINGER_ID_LEFT_LITTLE       (10)
#define UCBioAPI_FINGER_ID_MAX               (11)
```

UCBioAPI_FINGER_ID_UNKNOWN은 지문 Capture시 특정 손가락 정보를 알 수 없는 경우 사용 된다.

■ UCBioAPI_MATCH_OPTION_0

Prototype:

```
typedef struct ucbioapi_match_option_0 {
   UCBioAPI_UINT8         StructureType;
   UCBioAPI_UINT8         NoMatchFinger[UCBioAPI_FINGER_ID_MAX];
   UCBioAPI_UINT32        Reserved[8];
} UCBioAPI_MATCH_OPTION_0, *UCBioAPI_MATCH_OPTION_PTR_0;
```

Description:

UCBioAPI_VerifyMatchEx 함수에서 세밀한 인증을 위해 사용되는 정보를 담기 위한 구조체.

StructureType:

구조체 Type. 향후 추가되는 초기 설정 값이 있을 경우 구조체 1, 2, .. 등과 같이 증가될 수 있다. 현재는 0만 사용된다.

NoMatchFinger:

인증에 사용하지 않을 손가락 ID를 지정 할 수 있다. FIR에 10개의 손가락이 모두 등록되어 있다 하더라도 그 중에 몇 개의 손가락 데이터는 인증에서 제외시키고자 할 경우 지정할 수 있다. 또한 각 손가락의 Sample 별로도 인증에서 제외시킬 수 있다.
1을 지정하면 첫 번째 Sample을 제외하고 2를 지정하면 두 번째 Sample을 제외하고 3을 지정하면 그 손가락은 인증에 사용하지 않게 된다.
만약 오른쪽 엄지 손가락의 첫 번째 Sample과 왼쪽 엄지손가락의 두 번째 Sample을 제외시키고자 한다면 다음과 같이 하면 된다.

```
NoMatchFinger[UCBioAPI_FINGER_ID_RIGHT_THUMB] = 1;
NoMatchFinger[UCBioAPI_FINGER_ID_LEFT_THUMB] = 2;
```

Reserved:

예약된 값

5.1.3. Export/Import functions related types

UCBioAPI_ExportType.h에 선언되어 있으며 Export 및 Import와 같은 데이터 변환과 관련한 Type 들에 대해 정의하고 있다.

■ **UCBioAPI_TEMPLATE_BLOCK**

Prototype:

```
typedef struct ucbioapi_template_block {
   UCBioAPI_UINT32        Length;
   UCBioAPI_UINT8*        Data;
} UCBioAPI_TEMPLATE_BLOCK, *UCBioAPI_TEMPLATE_BLOCK_PTR;
```

Description:

1개의 Template 정보를 담기 위한 구조체.

Length:

Data의 길이 값. 다른 구조체들과 달리 Structure의 길이 값이 아닌 것에 주의한다.

Data:

Template 데이터가 담기는 Binary 데이터 블록. 담겨진 데이터는 UCBioAPI_EXPORT_DATA 구조체의 TemplateType의 값에 따라 달라지게 된다.

■ **UCBioAPI_FINGER_BLOCK**

Prototype:

```
typedef struct ucbioapi_finger_block {
   UCBioAPI_UINT32               Length;
   UCBioAPI_FINGER_ID            FingerID;
   UCBioAPI_TEMPLATE_BLOCK_PTR   TemplateInfo;
} UCBioAPI_FINGER_BLOCK, *UCBioAPI_FINGER_BLOCK_PTR;
```

Description:

1개 손가락의 정보를 담기 위한 구조체. 1개의 손가락 안에는 여러 개의 Template 정보가 담길 수 있다.

Length:

구조체의 길이 값. sizeof(UCBioAPI_FINGER_BLOCK) 값을 가진다.

FingerID:

손가락 ID 값을 가진다. UCBioAPI_FINGER_ID 참조.

TemplateInfo:

Template 정보를 담는 구조체 배열.

다수의 Template 정보를 담기 위해 UCBioAPI_TEMPLATE_BLOCK이 여러 개의 배열로 존재

할 수 있다. 배열의 개수는 UCBioAPI_EXPORT_DATA 구조체의 SamplesPerFinger 값에 담겨 있다.

■ **UCBioAPI_EXPORT_DATA**

Prototype:

```
typedef struct ucbioapi_export_data {
   UCBioAPI_UINT32              Length;
   UCBioAPI_TEMPLATE_TYPE       TemplateType;
   UCBioAPI_UINT8               FingerNum;
   UCBioAPI_FINGER_ID           DefaultFingerID;
   UCBioAPI_UINT8               SamplesPerFinger;
   UCBioAPI_UINT8               Reserved;
   UCBioAPI_FINGER_BLOCK_PTR    FingerInfo;
} UCBioAPI_EXPORT_DATA, *UCBioAPI_EXPORT_DATA_PTR;
```

Description:

1개의 FIR에 대한 Template 정보를 담기 위한 구조체. 1개의 FIR 안에는 여러 개의 손가락 정보가 담길 수 있다.

Length:

구조체의 길이 값. sizeof(UCBioAPI_EXPORT_DATA) 값을 가진다.

TemplateType:

담겨진 Template의 데이터 Type. UCBioAPI_TEMPLATE_TYPE 참조.

FingerNum:

총 손가락의 개수를 지정. 여기에 지정된 개수만큼 FingerInfo 정보가 배열로 들어있다.

DefaultFingerID:

대표 손가락 ID를 지정.

SamplesPerFinger:

손가락당 Template의 개수를 지정. 여기에 지정된 개수만큼 FingerInfo 정보내의 TemplateInfo 정보가 배열로 들어있다.

Reserved:

예약된 값.

FingerInfo:

손가락 정보를 담는 구조체 배열.

다수의 손가락 정보를 담기 위해 UCBioAPI_FINGER_BLOCK이 여러 개의 배열로 존재 할 수 있다. 배열의 개수는 FingerNum 값에 담겨 있다.

■ **UCBioAPI_IMAGE_DATA**

Prototype:

```
typedef struct ucbioapi_image_data {
   UCBioAPI_UINT32        Length;
   UCBioAPI_UINT8*        Data;
} UCBioAPI_IMAGE_DATA, *UCBioAPI_IMAGE_DATA_PTR;
```

Description:
1개의 지문 이미지를 담기 위한 구조체.

Length:
구조체의 길이 값. sizeof(UCBioAPI_IMAGE_DATA) 값을 가진다.
Data:
지문 이미지 데이터가 담기는 Binary 데이터 블록. 이미지는 RAW 형식의 이미지 포맷으로 담겨 있으며 이 데이터 블록의 길이 값은 UCBioAPI_EXPORT_AUDIT_DATA 구조체의 ImageWidth값과 ImageHeight값을 곱한 값과 같다.
즉, length of Data = ImageWidth * ImageHeight 라고 보면 된다.

■ UCBioAPI_AUDIT_DATA

Prototype:

```
typedef struct ucbioapi_audit_data {
   UCBioAPI_UINT32              Length;
   UCBioAPI_UINT8               FingerID;
   UCBioAPI_IMAGE_DATA_PTR      Image;
} UCBioAPI_AUDIT_DATA, *UCBioAPI_AUDIT_DATA_PTR;
```

Description:
1개의 손가락에 대한 지문 이미지를 담기 위한 구조체. 1개의 손가락 안에는 여러 개의 지문 이미지 정보가 담길 수 있다.

Length:
구조체의 길이 값. sizeof(UCBioAPI_AUDIT_DATA) 값을 가진다.
FingerID:
손가락 ID 값을 가진다. UCBioAPI_FINGER_ID 참조.
Image:
지문 이미지 정보를 담는 구조체 배열.
다수의 지문 이미지 정보를 담기 위해 UCBioAPI_IMAGE_DATA가 여러 개의 배열로 존재할 수 있다. 배열의 개수는 UCBioAPI_EXPORT_AUDIT_DATA 구조체의 SamplesPerFinger 값에 담겨 있다.

■ UCBioAPI_EXPORT_AUDIT_DATA

Prototype:

```
typedef struct ucbioapi_export_audit_data {
   UCBioAPI_UINT32              Length;
   UCBioAPI_UINT8               FingerNum;
   UCBioAPI_UINT8               SamplesPerFinger;
   UCBioAPI_UINT32              ImageWidth;
   UCBioAPI_UINT32              ImageHeight;
   UCBioAPI_AUDIT_DATA_PTR      AuditData;
   UCBioAPI_UINT32              Reserved;
} UCBioAPI_EXPORT_AUDIT_DATA, *UCBioAPI_EXPORT_AUDIT_DATA_PTR;
```

Description:

1개의 Audit FIR에 대한 지문 이미지 정보를 담기 위한 구조체. 1개의 Audit FIR 안에는 여러 개의 손가락 이미지 정보가 담길 수 있다.

Length:

구조체의 길이 값. sizeof(UCBioAPI_EXPORT_AUDIT_DATA) 값을 가진다.

FingerNum:

총 손가락의 개수를 지정. 여기에 지정된 개수만큼 AuditData 정보가 배열로 들어있다.

SamplesPerFinger:

손가락당 지문 이미지의 개수를 지정. 여기에 지정된 개수만큼 AuditData 정보내의 Image 정보가 배열로 들어있다.

ImageWidth / ImageHeight:

이미지의 크기 값 지정.

AuditData:

손가락 이미지 정보를 담는 구조체 배열.

다수의 손가락 이미지 정보를 담기 위해 UCBioAPI_AUDIT_DATA가 여러 개의 배열로 존재할 수 있다. 배열의 개수는 FingerNum 값에 담겨 있다.

Reserved:

예약 된 값.

5.1.4. FastSearch functions related types

UCBioAPI_FastSearchType.h에 선언되어 있으며 FastSearch와 관련한 Type들에 대해 정의하고 있다.

■ **UCBioAPI_FASTSEARCH_INIT_INFO_0**

Prototype:

```
typedef struct ucbioapi_fastsearch_init_info_0 {
   UCBioAPI_UINT32              StructureType;
   UCBioAPI_UINT32              UseGroupMatch;
   UCBioAPI_UINT32              MatchMethod;
   UCBioAPI_UINT32              Researved1;
   UCBioAPI_UINT32              Researved2;
   UCBioAPI_UINT32              Researved3;
   UCBioAPI_UINT32              Researved4;
   UCBioAPI_UINT32              Researved5;
   UCBioAPI_UINT32*             Researved6;
} UCBioAPI_FASTSEARCH_INIT_INFO_0,
*UCBioAPI_FASTSEARCH_INIT_INFO_PTR_0;
```

Description:

FastSearch의 기본 초기 설정 값을 담는 구조체.
UCBioAPI_GetFastSearchInitInfo / UCBioAPI_SetFastSearchInitInfo 함수에서 사용된다. 각각의 값들에 대한 설명은 다음과 같다.

StructureType:

구조체 Type. 향후 추가되는 초기 설정 값이 있을 경우 구조체 1, 2, .. 등과 같이 증가될 수 있다. 현재는 0만 사용된다.

UseGroupMatch:

인증을 수행할 때 그룹단위의 인증을 할 것인지를 결정한다. 0일 경우 DB에 들어있는 순서대로 인증을 수행하고 1일 경우 그룹단위의 인증을 수행한다. 기본값은 1이다.
이 값은 가능한 1로 설정하고 인증을 하는 것이 좋다.

MatchMethod:

인증을 수행할 방법을 결정한다. 0일 경우 설정된 인증 레벨을 이용해 그 레벨을 넘는 것이 나오면 바로 인증을 종료하는 방법이고 1일 경우 최고점 인증 방법으로 가장 인증 레벨이 높은 값을 찾는 방식이다. 기본값은 0이다. 이 값은 가능한 0으로 설정하고 인증을 하는 것이 좋다.

Reserved1 ~ Reserved6:

예약 된 값.

■ **UCBioAPI_FASTSEARCH_INIT_INFO_0**

Prototype:

```
typedef struct ucbioapi_fastsearch_fp_info {
   UCBioAPI_UINT32        ID;
   UCBioAPI_UINT8         FingerID;
   UCBioAPI_UINT8         SampleNumber;
   UCBioAPI_UINT32        Reserved1;
   UCBioAPI_UINT32        Reserved2;
} UCBioAPI_FASTSEARCH_FP_INFO, *UCBioAPI_FASTSEARCH_FP_INFO_PTR;
```

Description:

FastSearch의 각 Template별 인증 정보를 담는 구조체. FastSearch로 인증 시 이 구조체에 값이 담겨 오게 된다. 각각의 값들에 대한 설명은 다음과 같다.

ID:

사용자 ID 값.

FingerID:

손가락 ID 값.

SampleNumber:

지문 Template 번호.

Reserved1 / Reserved2:

예약 된 값.

■ **UCBioAPI_FASTSEARCH_SAMPLE_INFO**

Prototype:

```
typedef struct ucbioapi_fastsearch_sample_info {
   UCBioAPI_UINT32        ID;
   UCBioAPI_UINT8         SampleCount[11];
} UCBioAPI_FASTSEARCH_SAMPLE_INFO,
*UCBioAPI_FASTSEARCH_SAMPLE_INFO_PTR;
```

Description:

FastSearch DB에 FIR을 추가하게 되면 추가된 FIR에 담겨진 각 Template의 정보를 얻을 수 있다. 그 정보를 담게 되는 구조체이다. UCBioAPI_AddFIRToFastSearchDB 함수에서 사용된다.

ID:

사용자 ID 값.

SamplesCount:

각 손가락 별로 Template의 개수가 담겨 있다. 1번 Index부터 오른손 엄지, 검지, 중지, 가

락지, 약지 순으로 증가하며 6번 Index부터 왼손 엄지, 검지, 중지, 가락지, 약지 순으로 증가한다. 0번 Index는 사용하지 않는다.
예를 들어 오른손 엄지에 담겨진 Template의 개수를 얻고 싶다면 다음과 같이 값을 얻을 수 있다.

```
int cnt = SampleCount[UCBioAPI_FINGER_ID_RIGHT_THUMB];
```

■ UCBioAPI_FASTSEARCH_CALLBACK_PARAM_0

Prototype:

```
typedef struct ucbioapi_fastsearch_callback_param_0 {
  UCBioAPI_UINT32              TotalCount;
  UCBioAPI_UINT32              MatchedIndex;
  UCBioAPI_UINT32              MatchedScore;
  UCBioAPI_UINT32              Reserved1;
  UCBioAPI_UINT32              Reserved2;
  UCBioAPI_UINT32              Reserved3;
  UCBioAPI_VOID_PTR            Reserved4;
} UCBioAPI_FASTSEARCH_CALLBACK_PARAM_0,
*UCBioAPI_FASTSEARCH_CALLBACK_PARAM_PTR_0;
```

Description:

FastSearch를 통해 1:N 인증을 수행 중에 매 인증이 수행될 때 마다 호출되는 Callback 함수의 첫 번째 인자로 넘어오는 구조체이다. 각종 정보를 담고 있다.

TotalCount:
현재 DB의 전체 지문 수.
MatchedIndex:
현재 인증 중인 지문의 Index 번호.
MatchedScore:
현재 인증 중인 지문의 인증 점수 값.
Reserved1 ~ Reserved4:
예약 된 값.

■ UCBioAPI_FASTSEARCH_CALLBACK_0

Prototype:

```
typedef UCBioAPI_RETURN (WINAPI* UCBioAPI_FASTSEARCH_CALLBACK_0)
(UCBioAPI_FASTSEARCH_CALLBACK_PARAM_PTR_0, UCBioAPI_VOID_PTR);
```

Description:

FastSearch를 통해 1:N 인증을 수행 중에 매 인증이 수행될 때 마다 호출되는 Callback 함수에 대한 함수 포인터 정의.

이 함수는 UCBioAPI_IdentifyFIRFromFastSearchDB 함수의 인자로 지정 할 수 있다. 첫 번째 인자로는 각 UCBioAPI_FASTSEARCH_CALLBACK_PARAM_0 구조체 정보가 넘어오고 두 번째 인자로는 사용자가 넘겨준 포인터의 값이 넘어온다.

Callback 함수는 0을 리턴하게 되는데 만약 0 이외의 값을 리턴하게 되면 현재 진행중인 FastSearch의 1:N인증을 중간에 종료시키게 된다.

■ UCBioAPI_FASTSEARCH_CALLBACK_INFO_0

Prototype:

```
typedef struct ucbioapi_fastsearch_callback_info_0 {
   UCBioAPI_UINT32                    CallBackType;
   UCBioAPI_FASTSEARCH_CALLBACK_0     CallBackFunction;
   UCBioAPI_VOID_PTR                  UserCallBackParam;
} UCBioAPI_FASTSEARCH_CALLBACK_INFO_0,
*UCBioAPI_FASTSEARCH_CALLBACK_INFO_PTR_0;
```

Description:

FastSearch의 Callback 함수를 지정하기 위한 구조체.

CallBackType:

Callback 함수의 Type을 지정. 이 값은 현재 항상 0의 값을 가진다.

CallBackFunction:

실제 Callback으로 불려질 함수를 지정.

UserCallBackParam:

Callback 함수의 두 번째 인자로 넘겨질 사용자 인자를 지정.

5.1.5. SmartCard functions related types

UCBioAPI_SmartCardType.h에 선언되어 있으며 스마트카드와 관련한 Type들에 대해 정의하고 있다.

■ **UCBioAPI_SC_USE_KEY_A / UCBioAPI_SC_USE_KEY_B**

Prototype:

```
#define UCBioAPI_SC_USE_KEY_A                (0x60)
#define UCBioAPI_SC_USE_KEY_B                (0x61)
```

Description:

Mifare 카드의 키 중에서 Key A를 사용 할 것인지 Key B를 사용 할 것인지를 지정 할 때 쓰는 값.

■ **UCBioAPI_SC_LED_TOGGLE / UCBioAPI_SC_LED_NOT_TOGGLE**

Prototype:

```
#define UCBioAPI_SC_LED_TOGGLE          (1)
#define UCBioAPI_SC_LED_NOT_TOGGLE      (0)
```

Description:

스마트카드 관련 함수 사용 시 함수 성공 및 실패 여부를 디바이스의 LED에도 표시 할 것인지 말 것인지를 지정한다. 만약 UCBioAPI_SC_LED_TOGGLE로 지정되면 스마트카드 관련 함수가 성공적으로 수행되면 디바이스의 LED가 파란색으로 변하고 그렇지 않다면 붉은색으로 변하게 된다.

5.2. Error definitions

UCBioBSP SDK에서 사용되는 각종 Error 값에 대한 정의와 그 Error 값에 대해 설명한다.
모든 Error 값은 UCBioAPI_Error.h 파일에 정의 되어 있다.

5.2.1. Success

성공 시 사용되는 Error 값에 대한 정의이다.

■ **UCBioAPIERROR_NONE**

Prototype:

```
#define UCBioAPIERROR_NONE                    (0)
```

Description:

성공 할 경우 가지는 Error 값. 이 경우 Error가 아니라 함수가 성공 했음을 의미한다.

5.2.2. General error definitions

일반적인 Error 값에 대한 정의이다.
이 Error 값은 UCBioAPIERROR_BASE_GENERAL(0) 값부터 시작한다.

■ **UCBioAPIERROR_INVALID_HANDLE**

Prototype:

```
#define UCBioAPIERROR_INVALID_HANDLE          (0x0001)
```

Description:

잘못된 Handle 값이 사용됨

■ **UCBioAPIERROR_INVALID_POINTER**

Prototype:

```
#define UCBioAPIERROR_INVALID_POINTER         (0x0002)
```

Description:

잘못된 포인터 값이 사용됨

■ **UCBioAPIERROR_INVALID_TYPE**

Prototype:

```
#define UCBioAPIERROR_INVALID_TYPE            (0x0003)
```

Description:

잘못된 Type 값이 사용됨. UCBioAPI_SetInitInfo 등과 같은 함수에서 지원하지 않는 StructureType을 함수 인자로 사용 할 경우 발생.

■ **UCBioAPIERROR_FUNCTION_FAIL**

Prototype:

```
#define UCBioAPIERROR_FUNCTION_FAIL           (0x0004)
```

Description:

함수 내부 오류가 발생해 함수 수행이 실패한 경우 발생.

■ **UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED**

Prototype:

```
#define UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED (0x0005)
```

Description:
StructureType과 일치하지 않는 Structure가 사용됨. UCBioAPI_SetInitInfo 등과 같은 함수에서 요청한 StructureType과 일치하지 않는 Structure가 인자로 사용 된 경우 발생.

■ **UCBioAPIERROR_ALREADY_PROCESSED**

Prototype:

```
#define UCBioAPIERROR_ALREADY_PROCESSED      (0x0006)
```

Description:
UCBioAPI_Process 함수에 입력한 FIR 데이터가 이미 Process 된 데이터 일 경우 발생.

■ **UCBioAPIERROR_EXTRACTION_OPEN_FAIL**

Prototype:

```
#define UCBioAPIERROR_EXTRACTION_OPEN_FAIL   (0x0007)
```

Description:
Extraction을 위한 Engine을 초기화 하는데 오류가 발생한 경우.

■ **UCBioAPIERROR_VERIFICATION_OPEN_FAIL**

Prototype:

```
#define UCBioAPIERROR_VERIFICATION_OPEN_FAIL (0x0008)
```

Description:
인증을 위한 Engine을 초기화 하는데 오류가 발생한 경우.

■ **UCBioAPIERROR_DATA_PROCESS_FAIL**

Prototype:

```
#define UCBioAPIERROR_DATA_PROCESS_FAIL      (0x0009)
```

Description:
지문 이미지 데이터로부터 특징점을 추출 하는 과정에서 오류가 발생.

■ **UCBioAPIERROR_MUST_BE_PROCESSED_DATA**

Prototype:

```
#define UCBioAPIERROR_MUST_BE_PROCESSED_DATA (0x000a)
```

Description:
특징점 데이터가 입력되어야 하는 함수에서 아직 Process 되지 않은 데이터가 입력된 경우 발생.

■ **UCBioAPIERROR_INTERNAL_CHECKSUM_FAIL**

Prototype:

```
#define UCBioAPIERROR_INTERNAL_CHECKSUM_FAIL (0x000b)
```

Description:
FIR 데이터의 내부 유효성 오류. 일반적으로 데이터가 임의로 변형되었거나 손상된 FIR 데이터가 사용된 경우 발생.

■ **UCBioAPIERROR_ENCRYPTED_DATA_ERROR**

Prototype:

```
#define UCBioAPIERROR_ENCRYPTED_DATA_ERROR   (0x000c)
```

Description:
FIR 데이터의 암호화 오류. 일반적으로 데이터가 임의로 변형되었거나 손상된 FIR 데이터가 사용되어 암호화를 해제 할 수 없을 경우 발생.

■ **UCBioAPIERROR_UNKNOWN_FORMAT**

Prototype:

```
#define UCBioAPIERROR_UNKNOWN_FORMAT         (0x000d)
```

Description:
알 수 없는 FIR Format 임.

■ **UCBioAPIERROR_UNKNOWN_VERSION**

Prototype:

```
#define UCBioAPIERROR_UNKNOWN_VERSION        (0x000e)
```

Description:
알 수 없는 FIR Version 임.

■ **UCBioAPIERROR_VALIDITY_FAIL**

Prototype:

```
#define UCBioAPIERROR_VALIDITY_FAIL          (0x000f)
```

Description:
UCBioBSP.dll 모듈의 유효성 오류. 일반적으로 DLL이 임의로 변형되었거나 Sign이 되지 않은 DLL을 사용하는 경우 발생.
UCBioBSP SDK는 DLL 스스로가 자신의 유효성을 검사하여 변형이 일어났는지를 검사하게 되어 있는데 외부에서 DLL을 1byte라도 조작하게 되면 이 오류가 발생하며 SDK는 동작하지 않게 되어 있다.

■ UCBioAPIERROR_INVALID_TEMPLATESIZE

Prototype:

```
#define UCBioAPIERROR_INVALID_TEMPLATESIZE   (0x0010)
```

Description:
FIR 데이터와 Template 데이터 간의 상호 변환 함수 사용 시 입력된 Template의 크기가 잘못된 경우 발생.

■ UCBioAPIERROR_INVALID_TEMPLATE

Prototype:

```
#define UCBioAPIERROR_INVALID_TEMPLATE       (0x0011)
```

Description:
잘못된 Template 데이터가 사용된 경우.

■ UCBioAPIERROR_EXPIRED_VERSION

Prototype:

```
#define UCBioAPIERROR_EXPIRED_VERSION        (0x0012)
```

Description:
UCBioBSP SDK를 Evaluation Version(평가판)으로 사용 할 경우 평가판의 사용 기간이 종료된 경우 발생.

■ UCBioAPIERROR_INVALID_SAMPLESPERFINGER

Prototype:

```
#define UCBioAPIERROR_INVALID_SAMPLESPERFINGER      (0x0013)
```

Description:

손가락당 Template의 개수가 잘못 지정 된 경우 발생.

■ **UCBioAPIERROR_UNKNOWN_INPUTFORMAT**

Prototype:

```
#define UCBioAPIERROR_UNKNOWN_INPUTFORMAT    (0x0014)
```

Description:

UCBioAPI_INPUT_FIR 구조체로 사용된 Format이 알 수 없는 Format인 경우 발생.

■ **UCBioAPIERROR_INVALID_PARAMETER**

Prototype:

```
#define UCBioAPIERROR_INVALID_PARAMETER      (0x0015)
```

Description:

함수 인자들 중 잘못된 인자가 사용되거나 범위를 넘는 인자 값이 사용된 경우 발생.

■ **UCBioAPIERROR_FUNCTION_NOT_SUPPORTED**

Prototype:

```
#define UCBioAPIERROR_FUNCTION_NOT_SUPPORTED (0x0016)
```

Description:

지원하지 않는 함수임.

5.2.3. Initialization related error definitions

초기화 값 설정에 관련된 Error 값에 대한 정의이다.
이 Error 값은 UCBioAPIERROR_BASE_INIT(0x0100) 값부터 시작한다.

■ **UCBioAPIERROR_INIT_MAXFINGERSFORENROLL**

Prototype:

```
#define UCBioAPIERROR_INIT_MAXFINGERSFORENROLL      (0x0101)
```

Description:

UCBioAPI_INIT_INFO_0의 MaxFingersForEnroll 값에 잘못된 값이 지정됨.

■ **UCBioAPIERROR_INIT_NECESSARYENROLLNUM**

Prototype:

```
#define UCBioAPIERROR_INIT_NECESSARYENROLLNUM       (0x0102)
```

Description:

UCBioAPI_INIT_INFO_0의 NecessaryEnrollNum 값에 잘못된 값이 지정됨.

■ **UCBioAPIERROR_INIT_SAMPLESPERFINGER**

Prototype:

```
#define UCBioAPIERROR_INIT_SAMPLESPERFINGER         (0x0103)
```

Description:

UCBioAPI_INIT_INFO_0의 SamplesPerFinger 값에 잘못된 값이 지정됨.

■ **UCBioAPIERROR_INIT_SECULEVELFORENROLL**
■ **UCBioAPIERROR_INIT_SECULEVELFORVERIFY**
■ **UCBioAPIERROR_INIT_SECULEVELFORIDENTIFY**

Prototype:

```
#define UCBioAPIERROR_INIT_SECULEVELFORENROLL       (0x0104)
#define UCBioAPIERROR_INIT_SECULEVELFORVERIFY       (0x0105)
#define UCBioAPIERROR_INIT_SECULEVELFORIDENTIFY     (0x0106)
```

Description:

UCBioAPI_INIT_INFO_0의 SecurityLevelForEnroll / SecurityLevelForVerify / SecurityLevelForIdentify 값에 잘못된 값이 지정됨.

5.2.4. Device related error definitions

디바이스와 관련된 Error 값에 대한 정의이다.
이 Error 값은 UCBioAPIERROR_BASE_DEVICE(0x0200) 값부터 시작한다.

■ **UCBioAPIERROR_DEVICE_OPEN_FAIL**

Prototype:

```
#define UCBioAPIERROR_DEVICE_OPEN_FAIL       (0x0201)
```

Description:

디바이스 초기화 실패. 디바이스가 없거나 디바이스 드라이버가 설치되지 않은 경우 발생.

■ **UCBioAPIERROR_INVALID_DEVICE_ID**

Prototype:

```
#define UCBioAPIERROR_INVALID_DEVICE_ID      (0x0202)
```

Description:

잘못된 디바이스 ID를 이용해 디바이스를 초기화 하려고 할 때 발생.

■ **UCBioAPIERROR_WRONG_DEVICE_ID**

Prototype:

```
#define UCBioAPIERROR_WRONG_DEVICE_ID        (0x0203)
```

Description:

현재 초기화 된 디바이스와 다른 디바이스 ID가 사용됨.

■ **UCBioAPIERROR_DEVICE_ALREADY_OPENED**

Prototype:

```
#define UCBioAPIERROR_DEVICE_ALREADY_OPENED  (0x0204)
```

Description:

디바이스가 이미 열려 있음. 디바이스를 두 번 초기화 할 경우 발생.

■ **UCBioAPIERROR_DEVICE_NOT_OPENED**

Prototype:

```
#define UCBioAPIERROR_DEVICE_NOT_OPENED      (0x0205)
```

Description:

디바이스가 초기화 되지 않은 상태로 디바이스 관련 함수를 사용 하려고 할 경우 발생.

■ **UCBioAPIERROR_DEVICE_BRIGHTNESS**
■ **UCBioAPIERROR_DEVICE_CONTRAST**
■ **UCBioAPIERROR_DEVICE_GAIN**

Prototype:

```
#define UCBioAPIERROR_DEVICE_BRIGHTNESS      (0x0206)
#define UCBioAPIERROR_DEVICE_CONTRAST        (0x0207)
#define UCBioAPIERROR_DEVICE_GAIN            (0x0208)
```

Description:

밝기 / 대비 / Gain 값에 대해 각각 잘못된 디바이스 설정 값이 사용됨.

5.2.5. User interface related error definitions

UI와 관련된 Error 값에 대한 정의이다.
이 Error 값은 UCBioAPIERROR_BASE_UI(0x0300) 값부터 시작한다.

■ **UCBioAPIERROR_USER_CANCEL**

Prototype:

```
#define UCBioAPIERROR_USER_CANCEL            (0x0301)
```

Description:

사용자가 Cancel 버튼을 눌러 취소 했음.

■ **UCBioAPIERROR_USER_BACK**

Prototype:

```
#define UCBioAPIERROR_USER_BACK              (0x0302)
```

Description:

사용자가 Back 버튼을 눌러 취소 했음. 현재는 이 오류 값은 사용되지 않음.

■ **UCBioAPIERROR_CAPTURE_TIMEOUT**

Prototype:

```
#define UCBioAPIERROR_CAPTURE_TIMEOUT        (0x0303)
```

Description:

지문 획득 중 시간 초과되어 종료됨.

■ **UCBioAPIERROR_CAPTURE_FAKE_SUSPICIOUS**

Prototype:

```
#define UCBioAPIERROR_CAPTURE_FAKE_SUSPICIOUS      (0x0304)
```

Description:

획득한 지문이 모조 지문으로 의심 됨. 현재는 이 오류 값은 사용되지 않음.

■ **UCBioAPIERROR_ENROLL_EVENT_PLACE**
■ **UCBioAPIERROR_ENROLL_EVENT_HOLD**
■ **UCBioAPIERROR_ENROLL_EVENT_REMOVE**
■ **UCBioAPIERROR_ENROLL_EVENT_PLACE_AGAIN**
■ **UCBioAPIERROR_ENROLL_EVENT_PROCESS**

■ **UCBioAPIERROR_ENROLL_EVENT_MATCH_FAILED**

Prototype:

```
#define UCBioAPIERROR_ENROLL_EVENT_PLACE            (0x0305)
#define UCBioAPIERROR_ENROLL_EVENT_HOLD             (0x0306)
#define UCBioAPIERROR_ENROLL_EVENT_REMOVE           (0x0307)
#define UCBioAPIERROR_ENROLL_EVENT_PLACE_AGAIN      (0x0308)
#define UCBioAPIERROR_ENROLL_EVENT_PROCESS          (0x0309)
#define UCBioAPIERROR_ENROLL_EVENT_MATCH_FAILED     (0x030a)
```

Description:

UCBioAPI_WINDOW_OPTION 구조체에서 FinishCallBackInfo에 Callback 함수를 등록 할 경우 그 Callback 함수의 첫 번째 인자로 오는 UCBioAPI_WINDOW_CALLBACK_PARAM_1 구조체의 dwResult 값으로 넘어올 수 있는 Event 값들이다. 이 값들은 지문 등록 과정에서만 Callback을 통해 넘어온다.

5.2.6. FastSearch related error definitions

FastSearch와 관련된 Error 값에 대한 정의이다.
이 Error 값은 UCBioAPIERROR_BASE_FASTSEARCH(0x0400) 값부터 시작한다.

■ **UCBioAPIERROR_FASTSEARCH_INIT_FAIL**

Prototype:

```
#define UCBioAPIERROR_FASTSEARCH_INIT_FAIL          (0x0401)
```

Description:

FastSearch Engine 초기화 실패.

■ **UCBioAPIERROR_FASTSEARCH_SAVE_DB**

Prototype:

```
#define UCBioAPIERROR_FASTSEARCH_SAVE_DB            (0x0402)
```

Description:

FastSearch용 DB 파일 저장 실패.

■ **UCBioAPIERROR_FASTSEARCH_LOAD_DB**

Prototype:

```
#define UCBioAPIERROR_FASTSEARCH_LOAD_DB            (0x0403)
```

Description:

FastSearch용 DB 파일 읽기 실패.

■ **UCBioAPIERROR_FASTSEARCH_UNKNOWN_VER**

Prototype:

```
#define UCBioAPIERROR_FASTSEARCH_UNKNOWN_VER        (0x0404)
```

Description:

FastSearch용 DB 파일이 알 수 없는 버전임.

■ **UCBioAPIERROR_FASTSEARCH_IDENTIFY_FAIL**

Prototype:

```
#define UCBioAPIERROR_FASTSEARCH_IDENTIFY_FAIL      (0x0405)
```

Description:

FastSearch Engine을 이용한 1:N 인증이 실패함.

■ **UCBioAPIERROR_FASTSEARCH_DUPLICATED_ID**

Prototype:

```
#define UCBioAPIERROR_FASTSEARCH_DUPLICATED_ID      (0x0406)
```

Description:

FastSearch DB에 FIR을 추가하려고 할 때 이미 DB에 동일한 ID의 사용자가 존재함.

■ **UCBioAPIERROR_FASTSEARCH_IDENTIFY_STOP**

Prototype:

```
#define UCBioAPIERROR_FASTSEARCH_IDENTIFY_STOP      (0x0407)
```

Description:

1:N 인증이 사용자에 의해 중단 됨. 일반적으로 FastSearch용 Callback 함수를 등록하고 인증 중 Callback 함수가 0 이외의 값을 리턴 할 경우 인증이 종료되고 이 오류 값을 리턴하게 된다.

■ **UCBioAPIERROR_FASTSEARCH_NOUSER_EXIST**

Prototype:

```
#define UCBioAPIERROR_FASTSEARCH_NOUSER_EXIST       (0x0408)
```

Description:

FastSearch용 DB에서 사용자를 찾거나 삭제하려고 할 경우 지정한 ID를 가진 사용자가 DB에 존재하지 않음.

5.2.7. Optional value related error definitions

UCBioAPI_FIR_OPTIONAL_DATA 구조체와 관련된 Error 값에 대한 정의이다.
이 Error 값은 UCBioAPIERROR_BASE_OPTIONAL(0x0500) 값부터 시작한다.

- **UCBioAPIERROR_OPTIONAL_UUID_FAIL**
- **UCBioAPIERROR_OPTIONAL_PIN1_FAIL**
- **UCBioAPIERROR_OPTIONAL_PIN2_FAIL**
- **UCBioAPIERROR_OPTIONAL_SITEID_FAIL**
- **UCBioAPIERROR_OPTIONAL_EXPIRE_DATE_FAIL**

Prototype:

```
#define UCBioAPIERROR_OPTIONAL_UUID_FAIL          (0x0501)
#define UCBioAPIERROR_OPTIONAL_PIN1_FAIL          (0x0502)
#define UCBioAPIERROR_OPTIONAL_PIN2_FAIL          (0x0503)
#define UCBioAPIERROR_OPTIONAL_SITEID_FAIL        (0x0504)
#define UCBioAPIERROR_OPTIONAL_EXPIRE_DATE_FAIL   (0x0505)
```

Description:

UCBioAPI_VerifyMatchEx 함수를 이용해 인증을 수행 할 때 추가 인증 데이터로 사용한 Optional Data에서 인증 실패 시 나오는 오류 값들이다.

UCBioAPIERROR_OPTIONAL_UUID_FAIL:
UCBioAPI_FIR_OPTIONAL_DATA 구조체의 UUID 값이 서로 일치하지 않음.

UCBioAPIERROR_OPTIONAL_PIN1_FAIL / UCBioAPIERROR_OPTIONAL_PIN2_FAIL:
UCBioAPI_FIR_OPTIONAL_DATA 구조체의 PIN1/PIN2 값이 서로 일치하지 않음.

UCBioAPIERROR_OPTIONAL_SITEID_FAIL:
UCBioAPI_FIR_OPTIONAL_DATA 구조체의 SiteID 값이 서로 일치하지 않음.

UCBioAPIERROR_OPTIONAL_EXPIRED_DATE_FAIL:
UCBioAPI_FIR_OPTIONAL_DATA 구조체의 ExpireDate의 기간이 초과됨.

5.2.8. SmartCard related error definitions

SmartCard와 관련된 Error 값에 대한 정의이다.
이 Error 값은 UCBioAPIERROR_BASE_SMARTCARD(0x0600) 값부터 시작한다.

■ **UCBioAPIERROR_SC_FUNCTION_FAILED**

Prototype:

```
#define UCBioAPIERROR_SC_FUNCTION_FAILED            (0x0601)
```

Description:

스마트카드 함수 수행 실패.

■ **UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE**

Prototype:

```
#define UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE       (0x0602)
```

Description:

현재 디바이스가 기능을 지원하지 않음.

■ **UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE**

Prototype:

```
#define UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE     (0x0603)
```

Description:

디바이스의 Firmware 버전이 낮아 기능을 지원하지 않음.

5.3. API References

UCBioBSP SDK에서 사용되는 각종 API 대한 정의와 그 함수 사용법 및 인자들에 대해 설명한다.

5.3.1. Basic API

기본적으로 사용된 API들에 대한 설명이다.
이 API들은 UCBioAPI.h 파일에 정의되어 있다.

■ **UCBioAPI_Init**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_Init (
            [OUT] UCBioAPI_HANDLE_PTR         phHandle);
```

Description:

UCBioAPI SDK를 사용하기 위해 초기화 하고 Handle 값을 얻어온다.

Parameters:

phHandle:
얻어올 UCBioAPI SDK의 Handle 값의 포인터.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_VALIDITY_FAIL
UCBioAPIERROR_EXPIRED_VERSION

■ **UCBioAPI_Terminate**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_Terminate (
            [IN] UCBioAPI_HANDLE              hHandle);
```

Description:

UCBioAPI SDK를 사용 종료하고 Handle 값을 닫는다.

Parameters:

hHandle:

종료할 UCBioAPI SDK의 Handle 값.

Returns:

UCBioAPIERROR_NONE

UCBioAPIERROR_INVALID_HANDLE

■ **UCBioAPI_GetVersion**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetVersion (
            [IN]  UCBioAPI_HANDLE            hHandle,
            [OUT] UCBioAPI_VERSION_PTR       pVersion);
```

Description:

UCBioAPI SDK의 버전 정보를 얻는다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

pVersion:

버전 정보를 담을 구조체 포인터.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER

■ **UCBioAPI_GetInitInfo**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetInitInfo (
            [IN]  UCBioAPI_HANDLE            hHandle,
            [IN]  UCBioAPI_UINT8             nStructureType,
            [OUT] UCBioAPI_INIT_INFO_PTR     pInitInfo);
```

Description:

UCBioAPI SDK의 초기 설정 값을 얻는다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nStructureType:

얻어오기를 원하는 정보 구조체의 Type 값. 이 값은 pInitInfo의 구조체 형태를 결정한다. 현재는 이 값이 0만을 지원한다.

pInitInfo:

얻어오기를 원하는 정보 구조체의 포인터. 반드시 nStructureType에서 지정한 구조체가 넘어가야 한다. 현재는 UCBioAPI_INIT_INFO_0 구조체만 지원하지만 향후 다른 구조체가 사용 될 수도 있다. pInitInfo 구조체의 StructureType의 값이 인자로 넘기기 전 반드시 두 번째 인자인 StructureType 값과 일치하도록 설정 한 뒤 이 함수를 호출해야 한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_INVALID_TYPE
UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED

■ **UCBioAPI_SetInitInfo**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SetInitInfo (
            [IN]  UCBioAPI_HANDLE            hHandle,
            [IN]  UCBioAPI_UINT8             nStructureType,
            [OUT] UCBioAPI_INIT_INFO_PTR     pInitInfo);
```

Description:

UCBioAPI SDK의 초기 설정 값을 재지정 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nStructureType:

설정하기를 원하는 정보 구조체의 Type 값. 이 값은 pInitInfo의 구조체 형태를 결정한다. 현재는 이 값이 0만을 지원한다.

pInitInfo:

설정하기를 원하는 정보 구조체의 포인터. 반드시 nStructureType에서 지정한 구조체가 넘어가야 한다. 현재는 UCBioAPI_INIT_INFO_0 구조체만 지원하지만 향후 다른 구조체가 사용 될 수도 있다. pInitInfo 구조체의 StructureType의 값이 인자로 넘기기 전 반드시 두 번째 인자인 StructureType 값과 일치하도록 설정 한 뒤 이 함수를 호출해야 한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_INVALID_TYPE
UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED
UCBioAPIERROR_INIT_MAXFINGERSFORENROLL
UCBioAPIERROR_INIT_NECESSARYENROLLNUM
UCBioAPIERROR_INIT_SAMPLESPERFINGER
UCBioAPIERROR_INIT_SECULEVELFORENROLL
UCBioAPIERROR_INIT_SECULEVELFORVERIFY
UCBioAPIERROR_INIT_SECULEVELFORIDENTIFY
UCBioAPIERROR_INIT_RESERVED1
UCBioAPIERROR_INIT_RESERVED2

■ **UCBioAPI_SetSkinResource**

Prototype:

```
UCBioAPI_BOOL UCBioAPI UCBioAPI_SetSkinResource (
                    [IN]  LPCTSTR               szResPath);
```

Description:

UCBioAPI SDK의 User Interface를 위한 Skin을 변경한다.
UCBioAPI SDK에서는 사용하는 등록 및 인증용 화면을 Skin 방식의 UI로 사용하고 있다. 때문에 UCBioBSP SDK에서 기본 제공하는 UI를 사용하지 않고 다른 언어로 된 UI나 또는 다른 형태의 UI를 사용하고자 하는 경우에는 사용자 정의 Skin을 만들어 사용 할 수 있다.

Parameters:

szResPath:
변경할 Skin Resource DLL의 Full path를 지정.

Returns:

UCBioAPIERROR_FALSE
UCBioAPIERROR_TRUE

■ **UCBioAPI_SetCaptureQuality**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SetCaptureQuality(
            [IN]  UCBioAPI_HANDLE           hHandle,
            [IN]  UCBioAPI_UINT32           nScanQuality,
            [IN]  UCBioAPI_UINT32           nEnrollQuality,
            [IN]  UCBioAPI_UINT32           nVerifyQuality,
            [IN]  UCBioAPI_UINT32           nIdentifyQuality);
```

Description:

UCBioAPI SDK의 입력 지문에 대한 품질 값을 설정한다. 지문 입력시 입력된 지문의 품질이 해당 설정값보다 이상인 경우만 입력지문으로 허용된다. 설정 가능값은 1부터100이며 기본값은 50이다. 설정값이 0인 경우 해당 설정값은 적용되지 않고 이전값이 유지된다.

Parameters:

nScanQuality : 입력된 지문의 순수한 이미지에 대한 품질의 최소 설정값
nEnrollQuality : 추출된 특징점 품질값에 대한 등록시 적용하는 최소 설정값
nVerifyQuality : 추출된 특징점 품질값에 대한 1:1매칭시 적용하는 최소 설정값
nIdentifyQuality : 추출된 특징점 품질값에 대한 1:N 매칭시 적용하는 최소 설정값
*** 현재 nScanQuality, nEnrollQuality 만 적용됨.**

Returns: UCBioAPIERROR_TRUE

5.3.2. Device related API

디바이스 관련된 API들에 대한 설명이다.
이 API들은 UCBioAPI.h 파일에 정의되어 있다.

■ **UCBioAPI_EnumerateDevice**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_EnumerateDevice (
            [IN] UCBioAPI_HANDLE              hHandle,
            [OUT] UCBioAPI_UINT32*            pNumDevice,
            [OUT] UCBioAPI_DEVICE_ID**        ppDeviceID,
            [OUT] UCBioAPI_DEVICE_INFO_EX**   ppDeviceInfoEx);
```

Description:

현재 시스템에 설치된 모든 디바이스를 검색해 목록을 만든다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pNumDevice:
검색된 디바이스의 개수를 담아올 값의 포인터.

ppDeviceID:
디바이스 ID의 목록에 대한 배열 포인터. 이 포인터 배열에 대한 메모리는 SDK 내부에서 할당되고 향후 사용이 종료되면 자동 해제되므로 함수 사용자는 메모리에 대해 할당하거나 해제 할 필요가 없다.

ppDeviceInfoEx:
디바이스에 대한 세부적인 정보를 담은 구조체의 배열 포인터. UCBioAPI_DEVICE_INFO_EX 구조체 참조. 이 포인터 배열에 대한 메모리는 SDK 내부에서 할당되고 향후 사용이 종료되면 자동 해제되므로 함수 사용자는 메모리에 대해 할당하거나 해제 할 필요가 없다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_DEVICE_OPEN_FAIL

■ **UCBioAPI_OpenDevice**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_OpenDevice (
            [IN]  UCBioAPI_HANDLE      hHandle,
            [IN]  UCBioAPI_DEVICE_ID   nDeviceID);
```

Description:

원하는 디바이스를 열고 초기화 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nDeviceID

열고자 하는 디바이스 ID 값. UCBioAPI_DEVICE_ID 참조.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_DEVICE_ID
UCBioAPIERROR_DEVICE_OPEN_FAIL
UCBioAPIERROR_DEVICE_ALREADY_OPENED

■ **UCBioAPI_CloseDevice**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_CloseDevice (
            [IN]  UCBioAPI_HANDLE      hHandle,
            [IN]  UCBioAPI_DEVICE_ID   nDeviceID);
```

Description:

열려진 디바이스를 닫고 사용을 종료한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nDeviceID

닫고자 하는 디바이스 ID 값. UCBioAPI_DEVICE_ID 참조.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_WRONG_DEVICE_ID

■ **UCBioAPI_GetDeviceInfo**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetDeviceInfo (
            [IN]  UCBioAPI_HANDLE             hHandle,
            [IN]  UCBioAPI_DEVICE_ID          nDeviceID,
            [IN]  UCBioAPI_UINT8              nStructureType,
            [OUT] UCBioAPI_DEVICE_INFO_PTR    pDeviceInfo);
```

Description:

현재 열려진 디바이스로부터 디바이스 설정 값을 읽어온다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nDeviceID

현재 열려진 디바이스 ID 값. UCBioAPI_DEVICE_ID 참조.

nStructureType:

얻어오기를 원하는 정보 구조체의 Type 값. 이 값은 pDeviceInfo의 구조체 형태를 결정한다. 현재는 이 값이 0만을 지원한다.

pDeviceInfo:

얻어오기를 원하는 정보 구조체의 포인터. 반드시 nStructureType에서 지정한 구조체가 넘어가야 한다. 현재는 UCBioAPI_DEVICE_INIT_INFO_0 구조체만 지원하지만 향후 다른 구조체가 사용 될 수도 있다. pDeviceInfo 구조체의 StructureType의 값이 인자로 넘기기 전 반드시 두 번째 인자인 StructureType 값과 일치하도록 설정 한 뒤 이 함수를 호출해야 한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_INVALID_TYPE
UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_WRONG_DEVICE_ID

■ **UCBioAPI_SetDeviceInfo**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SetDeviceInfo (
            [IN]  UCBioAPI_HANDLE             hHandle,
            [IN]  UCBioAPI_DEVICE_ID          nDeviceID,
            [IN]  UCBioAPI_UINT8              nStructureType,
            [OUT] UCBioAPI_DEVICE_INFO_PTR    pDeviceInfo);
```

Description:

현재 열려진 디바이스에 새로운 디바이스 설정 값을 재설정 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nDeviceID

현재 열려진 디바이스 ID 값. UCBioAPI_DEVICE_ID 참조.

nStructureType:

설정하기를 원하는 정보 구조체의 Type 값. 이 값은 pDeviceInfo의 구조체 형태를 결정한다. 현재는 이 값이 0만을 지원한다.

pDeviceInfo:

설정하기를 원하는 정보 구조체의 포인터. 반드시 nStructureType에서 지정한 구조체가 넘어가야 한다. 현재는 UCBioAPI_DEVICE_INIT_INFO_0 구조체만 지원하지만 향후 다른 구조체가 사용 될 수도 있다. pDeviceInfo 구조체의 StructureType의 값이 인자로 넘기기 전 반드시 두 번째 인자인 StructureType 값과 일치하도록 설정 한 뒤 이 함수를 호출해야 한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_INVALID_TYPE
UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_WRONG_DEVICE_ID
UCBioAPIERROR_DEVICE_BRIGHTNESS
UCBioAPIERROR_DEVICE_CONTRAST
UCBioAPIERROR_DEVICE_GAIN

■ **UCBioAPI_AdjustDevice**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_AdjustDevice (
            [IN]  UCBioAPI_HANDLE                   hHandle,
            [IN]  const UCBioAPI_WINDOW_OPTION_PTR  pWindowOption);
```

Description:

현재 열려진 디바이스의 밝기를 조정하는 UI를 띄운다. 하지만 현재 지원하는 디바이스는 내부적으로 자동 밝기 조절과정을 거치므로 이 함수를 사용 할 수 없다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

pWindowOption:

UI 설정을 위한 UCBioAPI_WINDOW_OPTION 구조체의 포인터. 이 값은 NULL 값을 가질 수 있으며 NULL로 지정하면 UI 설정을 디폴트로 사용한다.
자세한 설명은 UCBioAPI_WINDOW_OPTION 구조체 참조.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_USER_CANCEL

■ **UCBioAPI_GetOpenedDeviceID**

Prototype:

```
UCBioAPI_DEVICE_ID UCBioAPI UCBioAPI_GetOpenedDeviceID (
            [IN]  UCBioAPI_HANDLE      hHandle);
```

Description:

현재 열려진 디바이스 ID 값을 얻어온다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

Returns:

현재 열려진 디바이스 ID에 대한 UCBioAPI_DEVICE_ID 값.

■ **UCBioAPI_CheckFinger**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_AdjustDevice (
            [IN]  UCBioAPI_HANDLE      hHandle,
            [OUT] UCBioAPI_BOOL*       pbFingerExist);
```

Description:

현재 열려진 디바이스 위에 지문이 올려져 있는지를 검사해 그 결과를 알려준다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

pbFingerExist:

지문 존재 유무에 대한 값을 담는 포인터. 0 또는 1의 값이 담기게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED

■ **UCBioAPI_SetAutoDetect**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SetAutoDetect (
            [IN]  UCBioAPI_HANDLE      hHandle,
            [OUT] UCBioAPI_BOOL        bFlag);
```

Description:

지문 입력 시 자동 감지 기능의 유무를 설정한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

bFlag:

자동 감지 기능의 설정 유무(FALSE:기존 방식, TREU:설정 ➔ LED OFF상태 유지)

Returns:

UCBioAPIERROR_NONE

5.3.3. Memory related API

FIR 데이터나 메모리와 관련된 API들에 대한 설명이다.
이 API들은 UCBioAPI.h 파일에 정의되어 있다.
일반적으로 메모리 관련 함수들은 첫 번째 인자로 UCBioSDK용 Handle을 필요로 하지 않는다.

■ UCBioAPI_GetFIRFromHandle

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetFIRFromHandle (
            [IN]  UCBioAPI_HANDLE      hHandle,
            [OUT] UCBioAPI_FIR_PTR     pFIR);
```

Description:

FIR Handle로 부터 실제 FIR 데이터를 얻어온다. 이렇게 얻어진 FIR 데이터는 Binary 데이터를 포함하는 구조체이므로 Stream 데이터로 변환해 파일이나 DB에 저장하거나 네트워크를 통해 전송 할 수도 있다. 이렇게 얻어진 FIR 데이터는 반드시 사용이 끝난 후 UCBioAPI_FreeFIR 함수를 이용해 메모리 해제를 해 주어야 한다.

Parameters:

hHandle:
얻고자 하는 FIR Handle 값.

pFIR:
얻고자 하는 UCBioAPI_FIR 구조체의 포인터.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER

■ **UCBioAPI_GetExtendedFIRFromHandle**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetExtendedFIRFromHandle (
            [IN]  UCBioAPI_HANDLE      hHandle,
            [OUT] UCBioAPI_VOID_PTR    pFIR,
            [IN]  UCBioAPI_FIR_FORMAT  Format);
```

Description:

FIR Handle로 부터 실제 FIR 데이터를 얻어온다. UCBioAPI_GetFIRFromHandle 함수와 동일하지만 향후 추가되는 포맷의 FIR에 대해서도 얻어 올 수 있도록 Format 정보를 넘기는 것이 다르다. 이렇게 얻어진 FIR 데이터는 반드시 사용이 끝난 후 UCBioAPI_FreeFIR 함수를 이용해 메모리 해제를 해 주어야 한다.

Parameters:

hHandle:

얻고자 하는 FIR Handle 값.

pFIR:

얻고자 하는 UCBioAPI_FIR 구조체의 포인터. void* 이므로 향후 다른 포맷에 대해서도 대응이 가능하다.

Format:

얻고자 하는 FIR의 Format 값. 현재는 UCBioAPI_FIR_FORMAT_STANDARD(0) 만 지원한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_UNKNOWN_FORMAT

■ **UCBioAPI_GetHeaderFromHandle**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetHeaderFromHandle (
            [IN]  UCBioAPI_HANDLE            hHandle,
            [OUT] UCBioAPI_FIR_HEADER_PTR    pHeader);
```

Description:

FIR Handle로 부터 실제 FIR 데이터의 Header 정보만을 얻어온다.

Parameters:

hHandle:

얻고자 하는 FIR Handle 값.

pHeader:

얻고자 하는 UCBioAPI_FIR_HEADER 구조체의 포인터.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER

■ **UCBioAPI_GetExtendedFIRFromHandle**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetExtendedFIRFromHandle (
            [IN]  UCBioAPI_HANDLE      hHandle,
            [OUT] UCBioAPI_VOID_PTR    pFIR,
            [IN]  UCBioAPI_FIR_FORMAT  Format);
```

Description:

FIR Handle로 부터 실제 FIR 데이터의 Header 정보만을 얻어온다. UCBioAPI_GetHeaderFromHandle 함수와 동일하지만 향후 추가되는 포맷의 FIR에 대해서도 얻어 올 수 있도록 Format 정보를 넘기는 것이 다르다.

Parameters:

hHandle:

얻고자 하는 FIR Handle 값.

pHeader:

얻고자 하는 UCBioAPI_FIR_HEADER 구조체의 포인터. void* 이므로 향후 다른 포맷에 대해서도 대응이 가능하다.

Format:

얻고자 하는 FIR의 Format 값. 현재는 UCBioAPI_FIR_FORMAT_STANDARD(0) 만 지원한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_UNKNOWN_FORMAT

■ **UCBioAPI_GetTextFIRFromHandle**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetTextFIRFromHandle (
            [IN]  UCBioAPI_HANDLE             hHandle,
            [OUT] UCBioAPI_FIR_TEXTENCODE_PTR pTextFIR,
            [IN]  UCBioAPI_BOOL               bIsWide);
```

Description:

FIR Handle로 부터 실제 FIR 데이터를 Text String 형태로 얻어온다. 이렇게 얻어진 FIR 데이터는 String 형태의 데이터를 포함하는 구조체이므로 Stream 데이터로 변환해 파일이나 DB에 저장하거나 네트워크를 통해 전송 할 수도 있다. 이렇게 얻어진 FIR 데이터는 반드시 사용이 끝난 후 UCBioAPI_FreeTextFIR 함수를 이용해 메모리 해제를 해 주어야 한다.

Parameters:

hHandle:

얻고자 하는 FIR Handle 값.

pTextFIR:

얻고자 하는 UCBioAPI_FIR_TEXTENCODE 구조체의 포인터.

bIsWide:

얻고자 하는 Text String을 UNICODE로 할 것인지를 지정. UCBioAPI_TRUE로 지정할 경우 얻어지는 Text String은 UNICODE 형태로 담겨져 오게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER

■ **UCBioAPI_GetExtendedTextFIRFromHandle**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetExtendedTextFIRFromHandle (
            [IN]  UCBioAPI_HANDLE             hHandle,
            [OUT] UCBioAPI_FIR_TEXTENCODE_PTR pTextFIR,
            [IN]  UCBioAPI_BOOL               bIsWide,
            [IN]  UCBioAPI_FIR_FORMAT         Format);
```

Description:

FIR Handle로 부터 실제 FIR 데이터를 얻어온다. UCBioAPI_GetFIRFromHandle 함수와 동일하지만 향후 추가되는 포맷의 FIR에 대해서도 얻어 올 수 있도록 Format 정보를 넘기는 것이 다르다. 이렇게 얻어진 FIR 데이터는 반드시 사용이 끝난 후 UCBioAPI_FreeFIR 함수를 이용해 메모리 해제를 해 주어야 한다.

Parameters:

hHandle:

얻고자 하는 FIR Handle 값.

pTextFIR:

얻고자 하는 UCBioAPI_FIR_TEXTENCODE 구조체의 포인터.

bIsWide:

얻고자 하는 Text String을 UNICODE로 할 것인지를 지정. UCBioAPI_TRUE로 지정할 경우 얻어지는 Text String은 UNICODE 형태로 담겨져 오게 된다.

Format:

얻고자 하는 FIR의 Format 값. 현재는 UCBioAPI_FIR_FORMAT_STANDARD(0) 만 지원한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_UNKNOWN_FORMAT

■ **UCBioAPI_FreeFIRHandle**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_FreeFIRHandle (
                              [IN]  UCBioAPI_HANDLE  hHandle);
```

Description:

FIR Handle을 종료하고 메모리를 해제한다. FIR Handle의 값이 UCBioAPI_INVALID_HANDLE 일 경우에는 아무것도 하지 않는다.

Parameters:

hHandle:

사용 종료하고자 하는 FIR Handle 값.

Returns:

UCBioAPIERROR_NONE

UCBioAPIERROR_INVALID_HANDLE

■ **UCBioAPI_FreeFIR**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_FreeFIR (
            [IN]  UCBioAPI_VOID_PTR    pFIR);
```

Description:

FIR 데이터를 사용 종료하고 메모리를 해제한다. pFIR의 값이 NULL일 경우에는 아무것도 하지 않는다.

Parameters:

pFIR:

사용 종료하고자 하는 FIR 구조체의 포인터. 향후 추가될 FIR Format을 지원하기 위해 void*로 되어 있다.

Returns:

UCBioAPIERROR_NONE

UCBioAPIERROR_UNKNOWN_FORMAT

■ **UCBioAPI_FreeTextFIR**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_FreeTextFIR (
            [IN]  UCBioAPI_FIR_TEXTENCODE_PTR        pTextFIR);
```

Description:

Text FIR 데이터를 사용 종료하고 메모리를 해제한다. pTextFIR의 값이 NULL일 경우에는 아무것도 하지 않는다.

Parameters:

pTextFIR:

사용 종료하고자 하는 Text FIR 구조체의 포인터.

Returns:

UCBioAPIERROR_NONE

■ **UCBioAPI_FreePayload**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_FreePayload (
            [IN]  UCBioAPI_FIR_PAYLOAD_PTR          pPayload);
```

Description:

Payload 데이터를 사용 종료하고 메모리를 해제한다. pPayload의 값이 NULL일 경우에는 아무것도 하지 않는다.

Parameters:

pPayload:

사용 종료하고자 하는 Payload 구조체의 포인터.

Returns:

UCBioAPIERROR_NONE

5.3.4. Core API

UCBioBSP SDK의 주요 함수들인 Core API들에 대한 설명이다.
이 API들은 UCBioAPI.h 파일에 정의되어 있다.

■ **UCBioAPI_Capture**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_Capture (
              [IN]  UCBioAPI_HANDLE                  hHandle,
              [IN]  UCBioAPI_FIR_PURPOSE             nPurpose,
              [OUT] UCBioAPI_FIR_HANDLE_PTR          phCapturedFIR,
              [IN]  UCBioAPI_SINT32                  nTimeout,
              [OUT] UCBioAPI_FIR_HANDLE_PTR          phAuditData,
              [IN]  const UCBioAPI_WINDOW_OPTION_PTR pWindowOption);
```

Description:

현재 열려진 디바이스로부터 1개의 지문을 획득하여 FIR Handle을 만든다. 이렇게 획득된 phCapturedFIR이나 phAuditData는 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리를 해제해 주어야 한다. 이 함수는 디바이스를 사용하므로 사용 전 반드시 디바이스가 Open 되어 있어야 한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

nPurpose:
FIR 데이터의 Purpose 값을 지정 할 수 있다. 가질 수 있는 값은 다음과 같다.

```
#define UCBioAPI_FIR_PURPOSE_VERIFY                         (0x01)
#define UCBioAPI_FIR_PURPOSE_IDENTIFY                       (0x02)
#define UCBioAPI_FIR_PURPOSE_ENROLL                         (0x03)
#define UCBioAPI_FIR_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY   (0x04)
#define UCBioAPI_FIR_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY (0x05)
#define UCBioAPI_FIR_PURPOSE_AUDIT                          (0x06)
#define UCBioAPI_FIR_PURPOSE_UPDATE                         (0x10)
```

각각의 값들은 FIR 데이터에 대한 참조용 값으로만 사용되고 인증 시에는 영향을 미치지 않는다. 만약 이 값을 ENROLL 관련 Purpose로 지정 할 경우 UCBioAPI_Enroll 함수와 동일하게 동작한다.
좀 더 자세한 설명은 UCBioAPI_FIR_PURPOSE 정의 참조.

phCapturedFIR:
획득된 지문 데이터의 FIR Handle값을 담을 포인터.

nTimeout:
지문 획득을 위해 대기하는 최대 시간. 단위는 millisecond이므로 1초의 경우에는 1,000이라고 지정해야 한다. 가질 수 있는 값으로는 다음과 같은 값을 추가적으로 가질 수 있다.

```
#define UCBioAPI_NO_TIMEOUT                 (0)
#define UCBioAPI_USE_DEFAULT_TIMEOUT        (-1)
#define UCBioAPI_CONTINUOUS_CAPTRUE         (-2)
```

0을 지정하면 Timeout 없이 계속 지문 입력을 대기한다. 대기 중에 지문이 입력되면 입력을 종료하고 리턴한다.
-1을 지정하면 UCBioAPI_INIT_INFO 구조체에 디폴트로 지정한 값만큼 대기한다.
-2를 지정하면 지문이 입력되더라도 종료하지 않고 계속해서 지문 입력을 받는다.

phAuditData:
획득된 지문 이미지 데이터의 FIR Handle값을 담을 포인터. 이 값은 NULL 값을 가질 수 있으며 NULL로 지정하면 값을 얻어오지 않는다.

pWindowOption:
UI 설정을 위한 UCBioAPI_WINDOW_OPTION 구조체의 포인터. 이 값은 NULL 값을 가질 수 있으며 NULL로 지정하면 UI 설정을 디폴트로 사용한다.
자세한 설명은 UCBioAPI_WINDOW_OPTION 구조체 참조.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_USER_CANCEL
UCBioAPIERROR_CAPTURE_TIMEOUT
UCBioAPIERROR_CAPTURE_FAKE_SUSPICIOUS

■ **UCBioAPI_Process**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_Process (
              [IN]  UCBioAPI_HANDLE                  hHandle,
              [IN]  const UCBioAPI_INPUT_FIR_PTR     piCapturedFIR,
              [OUT] UCBioAPI_FIR_HANDLE_PTR          phProcessedFIR);
```

Description:

Audit FIR로부터 지문 특징점을 추출해 FIR Handle을 만든다. 이렇게 획득된 phProcessedFIR은 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리를 해제해 주어야 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

piCapturedFIR:

특징점 추출을 위한 FIR 데이터가 담겨진 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

phProcessedFIR:

획득된 지문 데이터의 FIR Handle값을 담을 포인터.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_UNKNOWN_INPUTFORMAT
UCBioAPIERROR_ALREADY_PROCESSED
UCBioAPIERROR_DATA_PROCESS_FAIL

■ **UCBioAPI_CreateTemplate**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_CreateTemplate (
              [IN]  UCBioAPI_HANDLE                  hHandle,
              [IN]  const UCBioAPI_INPUT_FIR_PTR     piCapturedFIR,
              [IN]  const UCBioAPI_INPUT_FIR_PTR     piStoredFIR,
              [OUT] UCBioAPI_FIR_HANDLE_PTR          phNewFIR,
              [IN]  const UCBioAPI_FIR_PAYLOAD_PTR   pPayload);
```

Description:

기존의 FIR 데이터에 새로운 FIR 데이터를 Merge하거나 새로운 FIR 데이터로 교체하여 새로운 FIR Handle을 만든다. 또한 새로운 Payload를 기존 FIR에 교체 할 경우에도 사용 가능하다. 이렇게 획득된 phNewFIR은 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리를 해제해 주어야 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

piCapturedFIR:

새롭게 교체 할 FIR 데이터가 담겨진 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

piStoredFIR:

베이스로 사용 할 FIR 데이터가 담겨진 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.
만약 이 값이 NULL이 아니라면 이 값을 베이스로 해서 FIR 데이터가 만들어진다.
즉, 만약 이 FIR 값이 오른손 엄지, 검지, 중지의 데이터를 가지고 있고 새롭게 교체할 piCapturedFIR 데이터가 오른손 엄지와 왼손 엄지 데이터를 가지고 있다면 최종적으로 만들어지는 FIR 데이터는 piStoredFIR 데이터에 있던 오른손 검지, 중지와 piCapturedFIR 데이터에 있던 오른손 엄지, 왼손 엄지 데이터를 모두 가지는 데이터로 만들어 진다. 즉, 기존 데이터에서 새로운 데이터가 추가 및 변경되어 새로운 데이터로 만들어 지게 된다.

phNewFIR:

획득된 지문 데이터의 FIR Handle값을 담을 포인터.

pPayload:

만들어질 phNewFIR 데이터에 담겨질 Payload 데이터가 담긴 구조체의 포인터.
이 값은 NULL이 사용 되어 질 수 있고 NULL이 지정되면 기존 데이터의 Payload 값이 유지된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_MUST_BE_PROCESSED_DATA
UCBioAPIERROR_EXTRACTION_OPEN_FAIL
UCBioAPIERROR_UNKNOWN_FORMAT

■ **UCBioAPI_VerifyMatch**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_VerifyMatch (
              [IN]  UCBioAPI_HANDLE                  hHandle,
              [IN]  const UCBioAPI_INPUT_FIR_PTR     piProcessedFIR,
              [IN]  const UCBioAPI_INPUT_FIR_PTR     piStoredFIR,
              [OUT] UCBioAPI_BOOL*                   pbResult,
              [IN]  const UCBioAPI_FIR_PAYLOAD_PTR   pPayload);
```

Description:

기존에 획득된 두 개의 FIR 데이터를 서로 비교하여 그 인증 결과를 얻어온다. 인증이 성공 할 경우 등록용 FIR 데이터에 들어있던 Payload 값을 얻어 올 수도 있다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

piProcessedFIR:

인증을 요하는 FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

piStoredFIR:

인증을 요하는 등록용 FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

pbResult:

인증 결과값을 담을 포인터. 0 또는 1의 값이 담겨 온다.

pPayload:

인증이 성공할 경우 piStoredFIR에 들어있던 Payload 값을 얻어 올 수 있는데 그 값을 담아올 구조체의 포인터를 지정한다. 이 값은 NULL이 사용 되어 질 수 있고 NULL이 지정되면 Payload 값을 얻어오지 않는다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_MUST_BE_PROCESSED_DATA
UCBioAPIERROR_EXTRACTION_OPEN_FAIL
UCBioAPIERROR_UNKNOWN_FORMAT

■ **UCBioAPI_VerifyMatchEx**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_VerifyMatchEx (
              [IN]  UCBioAPI_HANDLE                  hHandle,
              [IN]  const UCBioAPI_INPUT_FIR_PTR     piProcessedFIR,
              [IN]  const UCBioAPI_INPUT_FIR_PTR     piStoredFIR,
              [OUT] UCBioAPI_BOOL*                   pbResult,
              [IN]  const UCBioAPI_FIR_PAYLOAD_PTR   pPayload,
              [IN]  UCBioAPI_MATCH_OPTION_PTR        pMatchOption);
```

Description:

기존에 획득된 두 개의 FIR 데이터를 서로 비교하여 그 인증 결과를 얻어온다. 인증이 성공 할 경우 등록용 FIR 데이터에 들어있던 Payload 값을 얻어 올 수도 있다.
UCBioAPI_VerifyMatch 함수와 동일하지만 추가적인 인증 데이터를 인증에 사용 할 수 있는 값들을 지정 할 수 있다. 이 값들을 지정 할 경우 지문 인증 전에 이 값들에 대한 인증이 성공해야만 지문 인증을 거칠 수 있다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

piProcessedFIR:
인증을 요하는 FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

piStoredFIR:
인증을 요하는 등록용 FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

pbResult:
인증 결과값을 담을 포인터. 0 또는 1의 값이 담겨 온다.

pPayload:
인증이 성공할 경우 piStoredFIR에 들어있던 Payload 값을 얻어 올 수 있는데 그 값을 담아올 구조체의 포인터를 지정한다. 이 값은 NULL이 사용 되어 질 수 있고 NULL이 지정되면 Payload 값을 얻어오지 않는다.

pMatchOption:
추가적인 인증 데이터를 인증에 사용 할 수 있는 값을 지정한다. 자세한 값들에 대한 설명은 UCBioAPI_MATCH_OPTION 구조체 참조.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_MUST_BE_PROCESSED_DATA
UCBioAPIERROR_EXTRACTION_OPEN_FAIL
UCBioAPIERROR_UNKNOWN_FORMAT
UCBioAPIERROR_OPTIONAL_UUID_FAIL
UCBioAPIERROR_OPTIONAL_PIN1_FAIL
UCBioAPIERROR_OPTIONAL_PIN2_FAIL
UCBioAPIERROR_OPTIONAL_SITEID_FAIL
UCBioAPIERROR_OPTIONAL_EXPIRE_DATE_FAIL

■ **UCBioAPI_Enroll**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_Enroll (
                [IN]  UCBioAPI_HANDLE                   hHandle,
                [IN]  const UCBioAPI_INPUT_FIR_PTR      piStoredFIR,
                [OUT] UCBioAPI_FIR_HANDLE_PTR           phEnrolledFIR,
                [IN]  const UCBioAPI_FIR_PAYLOAD_PTR    pPayload,
                [IN]  UCBioAPI_SINT32                   nTimeout,
                [OUT] UCBioAPI_FIR_HANDLE_PTR           phAuditData,
                [IN]  const UCBioAPI_WINDOW_OPTION_PTR pWindowOption);
```

Description:

현재 열려진 디바이스로부터 1명의 지문을 획득하여 FIR Handle을 만든다. 1명의 지문이기 때문에 다수개의 손가락에 대한 지문을 한번에 등록 받아 하나의 FIR Handle로 만들게 된다. 이렇게 획득된 phEnrolledFIR이나 phAuditData는 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리를 해제해 주어야 한다. 이 함수는 디바이스를 사용하므로 사용 전 반드시 디바이스가 Open 되어 있어야 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

piStoredFIR:

기존 FIR 데이터를 수정 할 경우 이 값을 지정한다. 이 값을 NULL로 지정 할 경우 새로운 지문을 입력 받게 된다.

phEnrolledFIR:

등록된 지문 데이터의 FIR Handle값을 담을 포인터.

pPayload:

만들어질 phNewFIR 데이터에 담겨질 Payload 데이터가 담긴 구조체의 포인터.
이 값은 NULL이 사용 되어 질 수 있고 NULL이 지정되면 기존 데이터의 Payload 값이 유지된다.

nTimeout:

지문 획득을 위해 대기하는 최대 시간. 단위는 millisecond이므로 1초의 경우에는 1,000이라고 지정해야 한다. 가질 수 있는 값으로는 다음과 같은 값을 추가적으로 가질 수 있다.

```
#define UCBioAPI_NO_TIMEOUT                 (0)
#define UCBioAPI_USE_DEFAULT_TIMEOUT        (-1)
```

0을 지정하면 Timeout 없이 계속 지문 입력을 대기한다. 대기 중에 지문이 입력되면 입력이 종료되고 다음으로 넘어간다.
-1을 지정하면 UCBioAPI_INIT_INFO 구조체에 디폴트로 지정한 값만큼 대기한다.
-2는 UCbioAPI_Enroll 함수에서는 사용하지 않는다.

phAuditData:
획득된 지문 이미지 데이터의 FIR Handle값을 담을 포인터. 이 값은 NULL 값을 가질 수 있으며 NULL로 지정하면 값을 얻어오지 않는다.

pWindowOption:
UI 설정을 위한 UCBioAPI_WINDOW_OPTION 구조체의 포인터. 이 값은 NULL 값을 가질 수 있으며 NULL로 지정하면 UI 설정을 디폴트로 사용한다.
자세한 설명은 UCBioAPI_WINDOW_OPTION 구조체 참조.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FUNCTION_FAIL
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_USER_CANCEL
UCBioAPIERROR_USER_BACK

■ **UCBioAPI_Verify**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_Verify (
                [IN]  UCBioAPI_HANDLE                   hHandle,
                [IN]  const UCBioAPI_INPUT_FIR_PTR      piStoredFIR,
                [OUT] UCBioAPI_BOOL*                    pbResult,
                [OUT] UCBioAPI_FIR_PAYLOAD_PTR          pPayload,
                [IN]  UCBioAPI_SINT32                   nTimeout,
                [OUT] UCBioAPI_FIR_HANDLE_PTR           phAuditData,
                [IN]  const UCBioAPI_WINDOW_OPTION_PTR pWindowOption);
```

Description:

기존에 획득된 FIR 데이터와 현재 디바이스로부터 라이브로 입력 받는 지문 데이터를 서로 비교하여 그 인증 결과를 얻어온다. 인증이 성공 할 경우 등록용 FIR 데이터에 들어있던 Payload 값을 얻어 올 수도 있다. UCBioAPI_VerifyMatch 함수와 거의 동일하지만 라이브 지문을 직접 입력 받아 인증을 수행하는 점이 다르다.
즉, 내부적으로 UCBioAPI_Capture 함수와 UCBioAPI_VerifyMatch 함수가 같이 수행되는 것으로 생각 할 수 있다. 이 함수는 디바이스를 사용하므로 사용 전 반드시 디바이스가 Open 되어 있어야 한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

piStoredFIR:
인증을 요하는 등록용 FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

pbResult:
인증 결과값을 담을 포인터. 0 또는 1의 값이 담겨 온다.

pPayload:
인증이 성공할 경우 piStoredFIR에 들어있던 Payload 값을 얻어 올 수 있는데 그 값을 담아올 구조체의 포인터를 지정한다. 이 값은 NULL이 사용 되어 질 수 있고 NULL이 지정되면 Payload 값을 얻어오지 않는다.

nTimeout:
지문 획득을 위해 대기하는 최대 시간. 단위는 millisecond이므로 1초의 경우에는 1,000이라고 지정해야 한다. 가질 수 있는 값으로는 다음과 같은 값을 추가적으로 가질 수 있다.

```
#define UCBioAPI_NO_TIMEOUT                     (0)
```

```
#define UCBioAPI_USE_DEFAULT_TIMEOUT          (-1)
#define UCBioAPI_CONTINUOUS_CAPTRUE           (-2)
```

0을 지정하면 Timeout 없이 계속 지문 입력을 대기한다. 대기 중에 지문이 입력되면 입력을 종료하고 리턴한다.
-1을 지정하면 UCBioAPI_INIT_INFO 구조체에 디폴트로 지정한 값만큼 대기한다.
-2를 지정하면 지문이 입력되더라도 종료하지 않고 계속해서 지문 입력을 받는다.

phAuditData:
획득된 지문 이미지 데이터의 FIR Handle값을 담을 포인터. 이 값은 NULL 값을 가질 수 있으며 NULL로 지정하면 값을 얻어오지 않는다.

pWindowOption:
UI 설정을 위한 UCBioAPI_WINDOW_OPTION 구조체의 포인터. 이 값은 NULL 값을 가질 수 있으며 NULL로 지정하면 UI 설정을 디폴트로 사용한다.
자세한 설명은 UCBioAPI_WINDOW_OPTION 구조체 참조.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_MUST_BE_PROCESSED_DATA
UCBioAPIERROR_EXTRACTION_OPEN_FAIL
UCBioAPIERROR_UNKNOWN_FORMAT
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_USER_CANCEL
UCBioAPIERROR_CAPTURE_TIMEOUT
UCBioAPIERROR_CAPTURE_FAKE_SUSPICIOUS

5.3.5. Data conversion API

데이터 변환과 관련된 API들에 대한 설명이다.
이 API들은 UCBioAPI_Export.h 파일에 정의되어 있다.

■ **UCBioAPI_FIRToTemplate**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_FIRToTemplate (
              [IN]  UCBioAPI_HANDLE                 hHandle,
              [IN]  const UCBioAPI_INPUT_FIR_PTR    piFIR,
              [OUT] UCBioAPI_EXPORT_DATA_PTR        pExportData,
              [IN]  UCBioAPI_TEMPLATE_TYPE          nExportType);
```

Description:

FIR 데이터로부터 원하는 형태의 Template 정보를 얻어 온다. 함수 호출 후 각각의 Template 정보가 담긴 구조체를 얻을 수 있다. 이렇게 얻어진 pExportData는 사용이 끝난 후 반드시 UCBioAPI_FreeExportData 함수를 이용해 메모리 해제를 해 주어야 한다.
일반적으로 FIR 데이터는 내부 데이터를 암호화 해 숨기고 있기 때문에 각 Template 별로 정보를 알 수는 없다. 때문에 다른 응용 프로그램이나 단말기와 같은 곳에서 Template 별로 데이터를 처리해야 하는 경우에는 이 함수를 이용해 데이터를 변환해야지만 사용 할 수 있다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

piFIR:
변환을 요하는 FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

pExportData:
변환된 Template 데이터가 담길 구조체의 포인터. UCBioAPI_EXPORT_DATA 구조체 참조.

nExportType:
변환 할 Template 데이터의 Type을 지정.
가질 수 있는 값은 UCBioAPI_TEMPLATE_TYPE 정의 참조.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER

UCBioAPIERROR_MUST_BE_PROCESSED_DATA
UCBioAPIERROR_FUNCTION_FAIL

■ **UCBioAPI_TemplateToFIR**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_TemplateToFIR (
                [IN]  UCBioAPI_HANDLE           hHandle,
                [IN]  UCBioAPI_UINT8*           pTemplateData,
                [IN]  UCBioAPI_UINT32           nTemplateDataSize,
                [IN]  UCBioAPI_UINT32           nTemplateDataType,
                [IN]  UCBioAPI_FIR_PURPOSE      nPurpose,
                [OUT] UCBioAPI_FIR_HANDLE_PTR   phProcessedFIR);
```

Description:

특정 형태를 가지는 1개의 Template 정보를 이용해 FIR Handle을 만든다. 함수 호출 후 Template 정보가 담긴 FIR Handle을 얻을 수 있다. 이렇게 얻어진 phProcessedFIR은 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리 해제를 해 주어야 한다. 일반적으로 UCBioBSP SDK는 Template을 이용해 바로 인증을 수행 할 수는 없도록 되어 있으며 모든 데이터는 FIR 형태로 변환을 거친 후 사용 할 수 있도록 되어 있다. 그러므로 이 함수를 이용해 FIR 데이터로 변환하는 작업이 필요하다.
이 함수는 Template 데이터를 간단하게 FIR Handle로 변환하기 위해 사용되며 만약 좀더 자세한 정보를 이용해 FIR Handle로 변환하고자 한다면 UCBioAPI_ImportDataToFIR 함수를 사용하도록 한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pTemplateData:
변환을 요하는 1개의 Template 데이터가 담긴 메모리 블록.

nTemplateDataSize:
변환 될 Template 데이터의 길이 값.

nTemplateDataType:
변환 될 Template 데이터의 Type을 지정.
가질 수 있는 값은 UCBioAPI_TEMPLATE_TYPE 정의 참조.

nPurpose:
변환 할 FIR 데이터의 Purpose 값을 지정 할 수 있다.
이 값들은 FIR 데이터에 대한 참조용 값으로만 사용되고 인증 시에는 영향을 미치지 않는다. 좀 더 자세한 설명은 UCBioAPI_FIR_PURPOSE 정의 참조.

phProcessedFIR:

변환 된 지문 데이터의 FIR Handle값을 담을 포인터.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FUNCTION_FAIL
UCBioAPIERROR_INVALID_TEMPLATESIZE

■ **UCBioAPI_TemplateToFIREx**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_TemplateToFIREx (
                [IN]  UCBioAPI_HANDLE           hHandle,
                [IN]  UCBioAPI_UINT8*           pTemplateData,
                [IN]  UCBioAPI_UINT32           nTemplateDataSize,
                [IN]  UCBioAPI_UINT32           nOneTemplateDataSize,
                [IN]  UCBioAPI_UINT32           nTemplateDataType,
                [IN]  UCBioAPI_FIR_PURPOSE      nPurpose,
                [OUT] UCBioAPI_FIR_HANDLE_PTR   phProcessedFIR);
```

Description:

특정 형태를 가지는 다수개의 Template 정보를 이용해 FIR Handle을 만든다. 함수 호출 후 Template 정보가 담긴 FIR Handle을 얻을 수 있다. 이렇게 얻어진 phProcessedFIR은 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리 해제를 해 주어야 한다. 일반적으로 UCBioBSP SDK는 Template을 이용해 바로 인증을 수행 할 수는 없도록 되어 있으며 모든 데이터는 FIR 형태로 변환을 거친 후 사용 할 수 있도록 되어 있다. 그러므로 이 함수를 이용해 FIR 데이터로 변환하는 작업이 필요하다.
이 함수는 Template 데이터를 간단하게 FIR Handle로 변환하기 위해 사용되며 만약 좀더 자세한 정보를 이용해 FIR Handle로 변환하고자 한다면 UCBioAPI_ImportDataToFIR 함수를 사용하도록 한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pTemplateData:
변환을 요하는 다수개의 Template 데이터가 담긴 메모리 블록.

nTemplateDataSize:
변환 될 Template 데이터의 전체 길이 값.
전체 데이터의 길이 값은 nOneTemplateDataSize의 배수가 되어야 한다. 만약 각각 400bytes짜리 2개의 Template 데이터를 이용해 FIR을 만들고자 할 경우라면 pTemplateData는 800bytes 메모리 블록으로 만들어 두 개의 데이터를 연속으로 담고 nTemplateDataSize는 800으로 지정하고 nOneTemplateDataSize는 400으로 지정하면 된다.

nOneTemplateDataSize:
변환 될 1개의 Template 데이터의 길이 값.

nTemplateDataType:
변환 될 Template 데이터의 Type을 지정.

가질 수 있는 값은 UCBioAPI_TEMPLATE_TYPE 정의 참조.

nPurpose:
변환 할 FIR 데이터의 Purpose 값을 지정 할 수 있다.
이 값들은 FIR 데이터에 대한 참조용 값으로만 사용되고 인증 시에는 영향을 미치지 않는다. 좀 더 자세한 설명은 UCBioAPI_FIR_PURPOSE 정의 참조.

phProcessedFIR:
변환 된 지문 데이터의 FIR Handle값을 담을 포인터.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FUNCTION_FAIL
UCBioAPIERROR_INVALID_TEMPLATESIZE

■ **UCBioAPI_ConvertTemplateData**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_ConvertTemplateData (
                [IN]  UCBioAPI_HANDLE          hHandle,
                [IN]  UCBioAPI_UINT8*          pTemplateData,
                [IN]  UCBioAPI_UINT32          nTemplateDataSize,
                [IN]  UCBioAPI_UINT32          nOneTemplateDataSize,
                [IN]  UCBioAPI_UINT32          nTemplateDataType,
                [IN]  UCBioAPI_UINT32          nConvertType,
                [OUT] UCBioAPI_UINT8**         ppConvertedData,
                [OUT] UCBioAPI_UINT32**        ppConvertedDataLen);
```

Description:

특정 형태를 가지는 다수개의 Template 정보를 이용해 다른 형태의 Template 정보로 변환할 경우에 사용한다. 함수 호출 후 Template 정보가 담긴 메모리 블록 포인터를 얻을 수 있다. 이렇게 얻어진 ppConvertedData와 ppConvertedDataLen은 사용이 끝난 후 반드시 UCBioAPI_FreeData 함수를 이용해 메모리 해제를 해 주어야 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

pTemplateData:

변환을 요하는 다수개의 Template 데이터가 담긴 메모리 블록.

nTemplateDataSize:

변환 될 Template 데이터의 전체 길이 값.

전체 데이터의 길이 값은 nOneTemplateDataSize의 배수가 되어야 한다. 만약 각각 400bytes짜리 2개의 Template 데이터를 이용해 FIR을 만들고자 할 경우라면 pTemplateData는 800bytes 메모리 블록으로 만들어 두 개의 데이터를 연속으로 담고 nTemplateDataSize는 800으로 지정하고 nOneTemplateDataSize는 400으로 지정하면 된다.

nOneTemplateDataSize:

변환 될 1개의 Template 데이터의 길이 값.

nTemplateDataType:

변환 될 Template 데이터의 Type을 지정.

가질 수 있는 값은 UCBioAPI_TEMPLATE_TYPE 정의 참조.

nConvertType:

변환 할 Template 데이터의 Type을 지정.

가질 수 있는 값은 UCBioAPI_TEMPLATE_TYPE 정의 참조.

ppConvertedData:
변환 할 Template 데이터가 담길 메모리 블록 포인터. 이 값은 내부적으로 메모리가 할당되어 오므로 사용이 끝나면 반드시 UCBioAPI_FreeData 함수를 이용해 메모리 해제를 해 주어야 한다.
다수의 Template이 변환 된 경우 이 메모리 블록에도 다수의 Template이 순서대로 담겨져 있으며 각각의 Template의 크기는 ppConvertedDataLen에 배열로 순서대로 담겨져 있다.

ppConvertedDataLen:
변환 할 Template 데이터의 길이 값이 담길 메모리 블록 포인터. 이 값은 내부적으로 메모리가 할당되어 오므로 사용이 끝나면 반드시 UCBioAPI_FreeData 함수를 이용해 메모리 해제를 해 주어야 한다.
이 값은 배열 형태로 되어 있어 다수의 Template이 변환 된 경우 순서대로 길이 값이 담겨져 있다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FUNCTION_FAIL
UCBioAPIERROR_INVALID_TEMPLATESIZE

■ **UCBioAPI_ImportDataToFIR**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_ImportDataToFIR (
              [IN]  UCBioAPI_HANDLE           hHandle,
              [IN]  UCBioAPI_EXPORT_DATA_PTR  pExportData,
              [IN]  UCBioAPI_FIR_PURPOSE      nPurpose,
              [OUT] UCBioAPI_FIR_HANDLE_PTR   phProcessedFIR);
```

Description:

특정 형태를 가지는 다수개의 Template 정보를 이용해 FIR Handle을 만든다. 함수 호출 후 Template 정보가 담긴 FIR Handle을 얻을 수 있다. 이렇게 얻어진 phProcessedFIR은 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리 해제를 해 주어야 한다. 일반적으로 UCBioBSP SDK는 Template을 이용해 바로 인증을 수행 할 수는 없도록 되어 있으며 모든 데이터는 FIR 형태로 변환을 거친 후 사용 할 수 있도록 되어 있다. 그러므로 이 함수를 이용해 FIR 데이터로 변환하는 작업이 필요하다.
이 함수는 UCBioAPI_TemplateToFIR 함수와 동일한 기능을 수행하지만 좀 더 세부적인 손가락 정보를 같이 포함해 FIR Handle로 변환 할 수 있는 점이 다르다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pExportData:
변환될 Template 데이터가 담긴 구조체의 포인터. UCBioAPI_EXPORT_DATA 구조체 참조.

nPurpose:
변환 할 FIR 데이터의 Purpose 값을 지정 할 수 있다.
이 값들은 FIR 데이터에 대한 참조용 값으로만 사용되고 인증 시에는 영향을 미치지 않는다. 좀 더 자세한 설명은 UCBioAPI_FIR_PURPOSE 정의 참조.

phProcessedFIR:
변환 된 지문 데이터의 FIR Handle값을 담을 포인터.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FUNCTION_FAIL
UCBioAPIERROR_INVALID_TEMPLATESIZE

■ **UCBioAPI_ImportDataToFIREx**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_ImportDataToFIREx (
                [IN]  UCBioAPI_HANDLE           hHandle,
                [IN]  UCBioAPI_EXPORT_DATA_PTR  pExportData,
                [IN]  UCBioAPI_FIR_PURPOSE      nPurpose,
                [IN]  UCBioAPI_FIR_DATA_TYPE    nDataType,
                [OUT] UCBioAPI_FIR_HANDLE_PTR   phProcessedFIR,
                [IN]  UCBioAPI_VOID_PTR         pReserved);
```

Description:

특정 형태를 가지는 다수개의 Template 정보를 이용해 FIR Handle을 만든다. 함수 호출 후 Template 정보가 담긴 FIR Handle을 얻을 수 있다. 이렇게 얻어진 phProcessedFIR은 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리 해제를 해 주어야 한다. 일반적으로 UCBioBSP SDK는 Template을 이용해 바로 인증을 수행 할 수는 없도록 되어 있으며 모든 데이터는 FIR 형태로 변환을 거친 후 사용 할 수 있도록 되어 있다. 그러므로 이 함수를 이용해 FIR 데이터로 변환하는 작업이 필요하다.
이 함수는 UCBioAPI_ImportDataToFIR 함수와 동일한 기능을 수행하지만 FIR의 Data Type 정보까지 지정하는 것이 가능하다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pExportData:
변환될 Template 데이터가 담긴 구조체의 포인터. UCBioAPI_EXPORT_DATA 구조체 참조.

nPurpose:
변환 할 FIR 데이터의 Purpose 값을 지정 할 수 있다.
이 값들은 FIR 데이터에 대한 참조용 값으로만 사용되고 인증 시에는 영향을 미치지 않는다. 좀 더 자세한 설명은 UCBioAPI_FIR_PURPOSE 정의 참조.

nDataType:
변환 할 FIR 데이터의 Data Type 값을 지정 할 수 있다.
이 값들에 대한 좀 더 자세한 설명은 UCBioAPI_FIR_DATA_TYPE 정의 참조.

phProcessedFIR:
변환 된 지문 데이터의 FIR Handle값을 담을 포인터.

pReserved:
예약 된 인자.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FUNCTION_FAIL
UCBioAPIERROR_INVALID_TEMPLATESIZE

■ **UCBioAPI_AuditFIRToImage**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_AuditFIRToImage (
              [IN]  UCBioAPI_HANDLE                  hHandle,
              [IN]  const UCBioAPI_INPUT_FIR_PTR     piAuditFIR,
              [OUT] UCBioAPI_EXPORT_AUDIT_DATA_PTR  pExportAuditData);
```

Description:

이미지 정보가 들어있는 Audit FIR 데이터로부터 원하는 RAW 포맷으로 된 이미지 정보를 얻어 온다. 함수 호출 후 각각의 이미지 정보가 담긴 구조체를 얻을 수 있다. 이렇게 얻어진 pExportAuditData는 사용이 끝난 후 반드시 UCBioAPI_FreeExportAuditData 함수를 이용해 메모리 해제를 해 주어야 한다.

일반적으로 Audit FIR 데이터는 내부 데이터를 암호화 해 숨기고 있기 때문에 각 손가락별로 이미지 정보를 알 수는 없다. 때문에 다른 응용 프로그램이나 단말기와 같은 곳에서 손가락별로 데이터를 처리해야 하는 경우에는 이 함수를 이용해 데이터를 변환해야지만 Audit FIR 안에 든 이미지를 사용 할 수 있다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

piAuditFIR:

변환을 요하는 Audit FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

pExportAuditData:

변환된 이미지 데이터가 담길 구조체의 포인터. UCBioAPI_EXPORT_AUDIT_DATA 구조체 참조.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_ALREADY_PROCESSED
UCBioAPIERROR_UNKNOWN_INPUTFORMAT
UCBioAPIERROR_FUNCTION_FAIL

■ **UCBioAPI_ImageToAuditFIR**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_ImportDataToFIR (
              [IN]  UCBioAPI_HANDLE                 hHandle,
              [IN]  UCBioAPI_EXPORT_AUDIT_DATA_PTR  pExportAuditData,
              [OUT] UCBioAPI_FIR_HANDLE_PTR         phAuditFIR);
```

Description:

RAW 포맷을 가지는 다수개의 이미지 정보를 이용해 Audit FIR Handle을 만든다. 함수 호출 후 이미지 정보가 담긴 Audit FIR Handle을 얻을 수 있다. 이렇게 얻어진 phAuditFIR은 사용이 끝난 후 반드시 UCBioAPI_FreeFIRHandle 함수를 이용해 메모리 해제를 해 주어야 한다.

일반적으로 UCBioBSP SDK는 이미지를 이용해 바로 인증을 수행 할 수는 없도록 되어 있으며 모든 데이터는 FIR 형태로 변환을 거친 후 사용 할 수 있도록 되어 있다. 그러므로 이 함수를 이용해 FIR 데이터로 변환하는 작업이 필요하다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

pExportAuditData:

변환될 이미지 데이터가 담긴 구조체의 포인터. UCBioAPI_EXPORT_AUDIT_DATA 구조체 참조.

phAuditFIR:

변환 된 지문 이미지 데이터의 FIR Handle값을 담을 포인터.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FUNCTION_FAIL

■ **UCBioAPI_FreeData**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_FreeData (
              [IN] UCBioAPI_UINT8*            pData);
```

Description:

UCBioAPI_ConvertTemplateData 함수에 의해 만들어진 데이터의 메모리를 해제한다.

Parameters:

pData:

해제 할 데이터 블록의 포인터. 만약 이 값이 NULL 이라면 아무것도 하지 않는다.

Returns:

UCBioAPIERROR_NONE

■ **UCBioAPI_FreeExportData**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_FreeExportData (
            [IN] UCBioAPI_EXPORT_DATA_PTR   pExportData);
```

Description:

UCBioAPI_FIRToTemplate 함수에 의해 만들어진 UCBioAPI_EXPORT_DATA 구조체의 메모리를 해제한다.

Parameters:

pExportData:

해제 할 UCBioAPI_EXPORT_DATA 구조체의 포인터. 만약 이 값이 NULL 이라면 아무것도 하지 않는다.

Returns:

UCBioAPIERROR_NONE

■ **UCBioAPI_FreeExportAuditData**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_FreeExportAuditData (
            [IN] UCBioAPI_EXPORT_AUDIT_DATA_PTR  pExportAuditData);
```

Description:

UCBioAPI_AuditFIRToImage 함수에 의해 만들어진 UCBioAPI_EXPORT_AUDIT_DATA 구조체의 메모리를 해제한다.

Parameters:

pExportAuditData:

해제 할 UCBioAPI_EXPORT_AUDIT_DATA 구조체의 포인터. 만약 이 값이 NULL 이라면 아무것도 하지 않는다.

Returns:

UCBioAPIERROR_NONE

5.3.6. FastSearch API

FastSearch와 관련된 API들에 대한 설명이다.
이 API들은 UCBioAPI_FastSearch.h 파일에 정의되어 있다.

■ **UCBioAPI_InitFastSearchEngine**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_InitFastSearchEngine (
              [IN]  UCBioAPI_HANDLE    hHandle);
```

Description:

FastSearch Engine을 초기화 한다.
FastSearch Engine을 사용하기 위해서는 반드시 이 함수를 이용해 초기화를 한 후 사용하여야 하며 사용이 끝나면 UCBioAPI_TerminateFastSearchEngine 함수를 이용해 사용 종료를 해 주어야 한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_VERIFICATION_OPEN_FAIL

■ **UCBioAPI_TerminateFastSearchEngine**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_TerminateFastSearchEngine (
              [IN]  UCBioAPI_HANDLE    hHandle);
```

Description:

FastSearch Engine을 사용 종료 한다.
FastSearch Engine의 사용이 끝나면 반드시 이 함수를 이용해 사용 종료를 해 주어야 한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FASTSEARCH_INIT_FAIL

■ **UCBioAPI_GetFastSearchInitInfo**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetFastSearchInitInfo (
              [IN]  UCBioAPI_HANDLE           hHandle,
              [IN]  UCBioAPI_UINT8            nStructureType,
              [OUT] UCBioAPI_INIT_INFO_PTR    pInitInfo);
```

Description:

FastSearch 엔진의 초기 설정 값을 얻는다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nStructureType:

얻어오기를 원하는 정보 구조체의 Type 값. 이 값은 pInitInfo의 구조체 형태를 결정한다. 현재는 이 값이 0만을 지원한다.

pInitInfo:

얻어오기를 원하는 정보 구조체의 포인터. 반드시 nStructureType에서 지정한 구조체가 넘어가야 한다. 현재는 UCBioAPI_FASTSEARCH_INIT_INFO_0 구조체만 지원하지만 향후 다른 구조체가 사용 될 수도 있다. pInitInfo 구조체의 StructureType의 값이 인자로 넘기기 전 반드시 두 번째 인자인 StructureType 값과 일치하도록 설정 한 뒤 이 함수를 호출해야 한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_INVALID_TYPE
UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED
UCBioAPIERROR_FASTSEARCH_INIT_FAIL

■ **UCBioAPI_SetFastSearchInitInfo**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SetFastSearchInitInfo (
            [IN]  UCBioAPI_HANDLE            hHandle,
            [IN]  UCBioAPI_UINT8             nStructureType,
            [OUT] UCBioAPI_INIT_INFO_PTR     pInitInfo);
```

Description:

FastSearch 엔진의 초기 설정 값을 재지정 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nStructureType:

설정하기를 원하는 정보 구조체의 Type 값. 이 값은 pInitInfo의 구조체 형태를 결정한다. 현재는 이 값이 0만을 지원한다.

pInitInfo:

설정하기를 원하는 정보 구조체의 포인터. 반드시 nStructureType에서 지정한 구조체가 넘어가야 한다. 현재는 UCBioAPI_FASTSEARCH_INIT_INFO_0 구조체만 지원하지만 향후 다른 구조체가 사용 될 수도 있다. pInitInfo 구조체의 StructureType의 값이 인자로 넘기기 전 반드시 두 번째 인자인 StructureType 값과 일치하도록 설정 한 뒤 이 함수를 호출해야 한다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_INVALID_TYPE
UCBioAPIERROR_STRUCTTYPE_NOT_MATCHED
UCBioAPIERROR_FASTSEARCH_INIT_FAIL

■ **UCBioAPI_AddFIRToFastSearchDB**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_AddFIRToFastSearchDB (
            [IN]  UCBioAPI_HANDLE                    hHandle,
            [IN]  const UCBioAPI_INPUT_FIR_PTR       pInputFIR,
            [IN]  UCBioAPI_UINT32                    nUserID,
            [OUT] UCBioAPI_FASTSEARCH_SAMPLE_INFO_PTR pSampleInfo);
```

Description:

FastSearch용 DB에 FIR 데이터를 추가 한다.
1:N 인증을 위해서는 우선 1:N을 수행 할 DB를 구축해야 하는데 이 함수를 이용해 그 DB를 만들게 된다.
또한 1:N 인증은 내부적으로 FIR 단위가 아니라 Template 단위로 동작하게 되어있다. 따라서 1개의 FIR을 추가하더라도 FIR 내부에 여러 개의 Template이 존재할 경우 DB에는 여러 개의 데이터가 추가되게 된다. 추가되는 Template에 대한 정보는 pSampleInfo 값을 통해 얻을 수 있다.
이렇게 만들어진 DB는 사용이 끝난 후 반드시 UCBioAPI_ClearFastSearchDB 함수를 통해 메모리 해제를 해 주어야 한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pInputFIR:
추가할 FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

nUserID:
추가할 FIR의 사용자 ID 값.

pSampleInfo:
DB에 추가된 FIR의 Template 정보에 대한 값을 얻어 올 수 있다. 만약 값을 얻을 필요가 없다면 NULL을 지정한다. UCBioAPI_FASTSEARCH_SAMPLE_INFO 구조체 참조.
이 값을 이용해 사용자 프로그램에서 추가된 Template에 대한 데이터 관리를 할 수도 있다. 자세한 사용 예는 SDK의 Sample중 UCBioBSP_FastSearchDemo 폴더를 참조하면 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL

UCBioAPIERROR_FASTSEARCH_DUPLICATED_ID
UCBioAPIERROR_FASTSEARCH_UNKNOWN_VER

■ **UCBioAPI_RemoveFpFromFastSearchDB**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_RemoveFpFromFastSearchDB (
            [IN]  UCBioAPI_HANDLE                   hHandle,
            [IN]  UCBioAPI_FASTSEARCH_FP_INFO_PTR   pFpInfo);
```

Description:

FastSearch용 DB에서 특정 데이터를 삭제한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

pFpInfo:

삭제할 Template 정보가 담긴 UCBioAPI_FASTSEARCH_FP_INFO 구조체의 포인터.
UCBioAPI_FASTSEARCH_FP_INFO 구조체 참조.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL
UCBioAPIERROR_FASTSEARCH_NOUSER_EXIST

■ **UCBioAPI_RemoveUserFromFastSearchDB**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_RemoveUserFromFastSearchDB (
            [IN]  UCBioAPI_HANDLE      hHandle,
            [IN]  UCBioAPI_UINT32      nUserID);
```

Description:

FastSearch용 DB에서 특정 사용자 ID용 데이터를 모두 삭제한다.
UCBioAPI_RemoveFpFromFastSearchDB 함수와 비슷하지만 동일 사용자 ID로 여러 개의 Template 데이터가 DB에 들어 있는 경우 모두 삭제하는 것이 다르다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

nUserID:
삭제할 사용자 ID 값.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL
UCBioAPIERROR_FASTSEARCH_NOUSER_EXIST

■ **UCBioAPI_IdentifyFIRFromFastSearchDB**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_IdentifyFIRFromFastSearchDB (
     [IN]  UCBioAPI_HANDLE                          hHandle,
     [IN]  const UCBioAPI_INPUT_FIR_PTR             pInputFIR,
     [IN]  UCBioAPI_FIR_SECURITY_LEVEL              nSecuLevel,
     [OUT] UCBioAPI_FASTSEARCH_FP_INFO_PTR          pFpInfo,
     [IN]  UCBioAPI_FASTSEARCH_CALLBACK_INFO_PTR_0  pCallbackInfo0);
```

Description:

FastSearch용 DB에서 특정 FIR과의 1:N 인증을 시도한다.
함수 호출 후 인증이 성공하면 인증된 Template의 정보를 얻을 수 있다. 1:N의 특성상 인증 시간은 매번 달라 질 수 있으며 시스템 속도와 메모리에 따라서도 달라 질 수 있다.
인증 중에 사용자가 인증 과정을 알고 싶거나 또는 인증을 임의로 중지하려면 Callback 함수를 등록해 사용 할 수 있다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pInputFIR:
인증할 FIR 데이터가 담긴 UCBioAPI_INPUT_FIR 구조체의 포인터.
UCBioAPI_INPUT_FIR 구조체 참조.

nSecuLevel:
인증 시 사용할 보안 수준을 지정.
가질 수 있는 값은 UCBioAPI_FIR_SECURITY_LEVEL 정의 참조.

pFpInfo:
인증이 성공할 경우 인증이 성공한 Template 데이터의 정보가 담겨 있다.
이 값을 통해 사용자 ID, 손가락 ID, Template 번호 등의 정보를 알 수 있게 된다.

pCallbackInfo0:
인증이 수행되는 동안 불려질 Callback 함수를 지정 할 수 있다.
Callback 함수에 대한 자세한 사항은 UCBioAPI_FASTSEARCH_CALLBACK_INFO_0 정의를 참조하면 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER

UCBioAPIERROR_FASTSEARCH_INIT_FAIL
UCBioAPIERROR_INVALID_PARAMETER
UCBioAPIERROR_FASTSEARCH_IDENTIFY_STOP
UCBioAPIERROR_FASTSEARCH_IDENTIFY_FAIL

■ **UCBioAPI_ClearFastSearchDB**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_ClearFastSearchDB (
            [IN]  UCBioAPI_HANDLE      hHandle);
```

Description:

FastSearch용 DB를 메모리 해제 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL

■ **UCBioAPI_SaveFastSearchDBToFile**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SaveFastSearchDBToFile (
            [IN]  UCBioAPI_HANDLE            hHandle,
            [IN]  const UCBioAPI_CHAR*       szFilepath);
```

Description:

FastSearch용 DB를 파일로 저장 한다. 이렇게 메모리 DB를 파일로 저장하게 되면 다음 번에 사용 시 UCBioAPI_LoadFastSearchDBFromFile 함수를 이용해 빠르게 로드해 사용 할 수 있게 된다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

szFilepath:

파일로 저장할 파일명의 Full path 값.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL
UCBioAPIERROR_FASTSEARCH_SAVE_DB

■ **UCBioAPI_LoadFastSearchDBToFile**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_LoadFastSearchDBToFile (
            [IN]  UCBioAPI_HANDLE            hHandle,
            [IN]  const UCBioAPI_CHAR*       szFilepath);
```

Description:

FastSearch용 DB를 파일로부터 메모리로 읽어온다.
이렇게 읽어올 수 있는 파일은 UCBioAPI_SaveFastSearchDBToFile 함수를 이용해 저장된 파일만 가능하다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

szFilepath:
불러 올 파일명의 Full path 값.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL
UCBioAPIERROR_FASTSEARCH_LOAD_DB

■ **UCBioAPI_GetFpCountFromFastSearchDB**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetFpCountFromFastSearchDB (
            [IN]  UCBioAPI_HANDLE      hHandle);
            [OUT] UCBioAPI_UINT32*     pDataCount);
```

Description:

FastSearch용 DB에 있는 Template의 개수를 얻어온다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

pDataCount:

얻어올 개수가 담길 값의 포인터.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL

■ **UCBioAPI_GetFpInfoFromFastSearchDB**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_GetFpInfoFromFastSearchDB (
            [IN]  UCBioAPI_HANDLE                  hHandle,
            [IN]  UCBioAPI_UINT32                  nDataIndex,
            [OUT] UCBioAPI_FASTSEARCH_FP_INFO_PTR  pFpInfo);
```

Description:

FastSearch용 DB에서 특정 Index에 있는 Template 정보를 얻어온다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

nDataIndex:

얻어올 Index 값.

pFpInfo:

Template 데이터의 정보가 담겨질 UCBioAPI_FASTSEARCH_FP_INFO 구조체의 포인터.
이 값을 통해 사용자 ID, 손가락 ID, Template 번호 등의 정보를 알 수 있게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL
UCBioAPIERROR_INVALID_PARAMETER

■ **UCBioAPI_CheckFpExistInFastSearchDB**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_CheckFpExistInFastSearchDB (
            [IN]  UCBioAPI_HANDLE                   hHandle,
            [IN]  UCBioAPI_FASTSEARCH_FP_INFO_PTR   pFpInfo,
            [OUT] UCBioAPI_BOOL*                    pExist);
```

Description:

FastSearch용 DB에 특정 Template 정보를 가진 데이터가 존재하는지를 검사한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

pFpInfo:

검사할 Template 데이터의 정보가 담겨진 구조체의 포인터.

pExist:

존재 유무를 얻어올 포인터. 존재하면 1이라는 값이 담겨져 온다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_INVALID_POINTER
UCBioAPIERROR_FASTSEARCH_INIT_FAIL

5.3.7. SmartCard API

스마트카드와 관련된 API들에 대한 설명이다.
이 API들은 UCBioAPI_SmartCard.h 파일에 정의되어 있다.

- **참고 - 스마트카드 사용을 위한 함수들은 디바이스의 Firmware 버전에 따라 지원하지 않는 함수가 있을 수 있다.**

■ UCBioAPI_SC_RFPowerOn

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_RFPowerOn (
              [IN]  UCBioAPI_HANDLE    hHandle,
              [IN]  UCBioAPI_UINT16    wLed);
```

Description:

5초 이내에 RF 영역의 RF Power를 ON 한다.
대부분의 스마트카드 API 들은 RF Power가 On이 된 상태에서 동작한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_RFPowerOff**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_RFPowerOff (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역의 RF Power를 OFF 한다.

Parameters:

hHandle:

UCBioAPI SDK의 Handle 값.

wLed:

성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_RFFunction**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_RFFunction (
              [IN]  UCBioAPI_HANDLE    hHandle,
              [IN]  UCBioAPI_UINT8*    pCmdBuff,
              [IN]  UCBioAPI_UINT16    nCmdLen,
              [OUT] UCBioAPI_UINT8*    pResultBuff,
              [OUT] UCBioAPI_UINT16*   nResultLen,
              [IN]  UCBioAPI_UINT16    wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 명령어를 전송하고 그 결과를 받아 온다.
반드시 RF Power가 ON이 되어 있어야 한다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pCmdBuff:
전송할 명령어가 담긴 버퍼 포인터.

nCmdLen:
전송할 명령어가 담긴 버퍼 포인터의 길이.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_ReadSerial**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_ReadSerial (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [OUT] UCBioAPI_UINT8*     pResultBuff,
              [OUT] UCBioAPI_UINT16*    nResultLen,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 해당 시리얼넘버를 얻어온다.
이 명령어는 내부에 RF Power를 ON 하는 기능이 있으므로 함수 호출 전에 UCBioAPI_SC_RFPowerOn 함수를 호출 할 필요가 없다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_ReadBlock**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_ReadBlock (
              [IN]  UCBioAPI_HANDLE    hHandle,
              [IN]  UCBioAPI_UINT8     AuthMode,
              [IN]  UCBioAPI_UINT8     SectorNum,
              [IN]  UCBioAPI_UINT8     BlockNum,
              [IN]  UCBioAPI_UINT8*    KeyValue,
              [OUT] UCBioAPI_UINT8*    pResultBuff,
              [OUT] UCBioAPI_UINT16*   nResultLen,
              [IN]  UCBioAPI_UINT16    wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록의 데이터를 읽어온다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
읽고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
읽고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

KeyValue:
사용할 6bytes 길이의 키 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:

성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_WriteBlock**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_WriteBlock (
              [IN]  UCBioAPI_HANDLE    hHandle,
              [IN]  UCBioAPI_UINT8     AuthMode,
              [IN]  UCBioAPI_UINT8     SectorNum,
              [IN]  UCBioAPI_UINT8     BlockNum,
              [IN]  UCBioAPI_UINT8*    KeyValue,
              [IN]  UCBioAPI_UINT8*    pData,
              [IN]  UCBioAPI_UINT16    wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록에 16bytes 길이의 데이터를 기록한다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
읽고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

KeyValue:
사용할 6bytes 길이의 키 값.

pData:
기록 할 16bytes의 데이터가 담긴 메모리 포인터.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_ReadSector**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_ReadSector (
              [IN]  UCBioAPI_HANDLE    hHandle,
              [IN]  UCBioAPI_UINT8     AuthMode,
              [IN]  UCBioAPI_UINT8     SectorNum,
              [IN]  UCBioAPI_UINT8*    KeyValue,
              [OUT] UCBioAPI_UINT8*    pResultBuff,
              [OUT] UCBioAPI_UINT16*   nResultLen,
              [IN]  UCBioAPI_UINT16    wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector의 데이터를 읽어온다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
읽고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

KeyValue:
사용할 6bytes 길이의 키 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_WriteSector**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_WriteSector (
                [IN]  UCBioAPI_HANDLE     hHandle,
                [IN]  UCBioAPI_UINT8      AuthMode,
                [IN]  UCBioAPI_UINT8      SectorNum,
                [IN]  UCBioAPI_UINT8*     KeyValue,
                [IN]  UCBioAPI_UINT8*     pData,
                [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector에 48bytes 길이의 데이터를 기록한다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

KeyValue:
사용할 6bytes 길이의 키 값.

pData:
기록 할 48bytes의 데이터가 담긴 메모리 포인터.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED

UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_ReadSectorFieldContent**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_ReadSectorFieldContent (
              [IN]  UCBioAPI_HANDLE    hHandle,
              [IN]  UCBioAPI_UINT8     AuthMode,
              [IN]  UCBioAPI_UINT8     StartSectorNum,
              [IN]  UCBioAPI_UINT8     EndSectorNum,
              [IN]  UCBioAPI_UINT8*    KeyValue,
              [OUT] UCBioAPI_UINT8*    pResultBuff,
              [OUT] UCBioAPI_UINT16*   nResultLen,
              [IN]  UCBioAPI_UINT16    wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector 구간 안의 모든 데이터를 읽어온다.
지정 가능한 구간은 최대 10개 Sector이며 최대 얻어 올 수 있는 데이터 길이는 480bytes 까지다. (48 Bytes * 10 Sectors = 480 Bytes)
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

StartSectorNum:
읽고자 하는 Sector의 시작 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

EndSectorNum:
읽고자 하는 Sector의 끝 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

KeyValue:
사용할 6bytes 길이의 키 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_WriteSectorFieldContent**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_WriteSectorFieldContent (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [IN]  UCBioAPI_UINT8      AuthMode,
              [IN]  UCBioAPI_UINT8      StartSectorNum,
              [IN]  UCBioAPI_UINT8      EndSectorNum,
              [IN]  UCBioAPI_UINT8*     KeyValue,
              [IN]  UCBioAPI_UINT8*     pData,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector 구간 안의 모든 데이터에 기록한다.
지정 가능한 구간은 최대 10개 Sector이며 최대 기록 할 수 있는 데이터 길이는 480bytes 까지다. (48 Bytes * 10 Sectors = 480 Bytes)
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

StartSectorNum:
쓰고자 하는 Sector의 시작 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

EndSectorNum:
쓰고자 하는 Sector의 끝 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

KeyValue:
사용할 6bytes 길이의 키 값.

pData:
기록 할 48bytes의 데이터가 담긴 메모리 포인터.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌

게 된다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_PreValue**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_PreValue (
                [IN]  UCBioAPI_HANDLE     hHandle,
                [IN]  UCBioAPI_UINT8      AuthMode,
                [IN]  UCBioAPI_UINT8      SectorNum,
                [IN]  UCBioAPI_UINT8      BlockNum,
                [IN]  UCBioAPI_UINT8*     KeyValue,
                [IN]  UCBioAPI_UINT8*     pData,
                [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록을 Value 모드로 전환하고 4bytes의 pData 값을 기록한다.
만약 기록하는 pData의 값이 0x00000000 이라면 함수 호출 후 해당 블록은 다음과 같이 값이 기록되게 된다. (0x00000000FFFFFFFF00000000FFFFFFFF00FF00FF)
이렇게 Value 모드로 변환된 블록은 UCBioAPI_SC_ReadValue, UCBioAPI_SC_IncrementValue 와 같은 Value 모드용 함수들을 사용 할 수 있게 된다.
Value 모드에 대한 자세한 사항은 이 문서의 SmartCard 사용하기를 참조하면 된다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
쓰고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

KeyValue:
사용할 6bytes 길이의 키 값.

pData:
기록 할 4bytes의 데이터가 담긴 메모리 포인터.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_ReadValue**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_ReadValue (
              [IN]  UCBioAPI_HANDLE    hHandle,
              [IN]  UCBioAPI_UINT8     AuthMode,
              [IN]  UCBioAPI_UINT8     SectorNum,
              [IN]  UCBioAPI_UINT8     BlockNum,
              [IN]  UCBioAPI_UINT8*    KeyValue,
              [OUT] UCBioAPI_UINT8*    pResultBuff,
              [OUT] UCBioAPI_UINT16*   nResultLen,
              [IN]  UCBioAPI_UINT16    wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록의 4bytes Value 데이터를 읽어온다.
반드시 RF Power가 ON이 되어 있어야 하고 지정한 블록이 Value 모드이어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
읽고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
읽고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

KeyValue:
사용할 6bytes 길이의 키 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:

성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_IncrementValue**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_IncrementValue (
                [IN]  UCBioAPI_HANDLE     hHandle,
                [IN]  UCBioAPI_UINT8      AuthMode,
                [IN]  UCBioAPI_UINT8      SectorNum,
                [IN]  UCBioAPI_UINT8      BlockNum,
                [IN]  UCBioAPI_UINT8*     KeyValue,
                [IN]  UCBioAPI_UINT8*     pData,
                [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록의 4bytes Value 데이터를 지정한 값만큼 증가시킨다.
반드시 RF Power가 ON이 되어 있어야 하고 지정한 블록이 Value 모드이어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
쓰고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

KeyValue:
사용할 6bytes 길이의 키 값.

pData:
증가 시킬 4bytes의 Value 데이터가 담긴 메모리 포인터.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_DecrementValue**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_DecrementValue (
                [IN]  UCBioAPI_HANDLE     hHandle,
                [IN]  UCBioAPI_UINT8      AuthMode,
                [IN]  UCBioAPI_UINT8      SectorNum,
                [IN]  UCBioAPI_UINT8      BlockNum,
                [IN]  UCBioAPI_UINT8*     KeyValue,
                [IN]  UCBioAPI_UINT8*     pData,
                [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록의 4bytes Value 데이터를 지정한 값만큼 감소시킨다.
반드시 RF Power가 ON이 되어 있어야 하고 지정한 블록이 Value 모드이어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
쓰고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

KeyValue:
사용할 6bytes 길이의 키 값.

pData:
감소 시킬 4bytes의 Value 데이터가 담긴 메모리 포인터.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_WriteSectorTrailer**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_WriteSectorTrailer (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [IN]  UCBioAPI_UINT8      AuthMode,
              [IN]  UCBioAPI_UINT8      SectorNum,
              [IN]  UCBioAPI_UINT8*     KeyValue,
              [IN]  UCBioAPI_UINT8*     NewAccessBit,
              [IN]  UCBioAPI_UINT8*     NewKeyA,
              [IN]  UCBioAPI_UINT8*     NewKeyB,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector의 Sector Trailer 영역을 수정한다.
Sector Trailer에는 해당 Sector의 Key A, Access Bits, Key B 값이 들어 있다.
해당 Sector의 Access 권한에 따라 성공, 실패 여부가 결정된다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 Mifare 카드 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

AuthMode:
Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정.
UCBioAPI_SC_USE_KEY_A(0x60) 또는 UCBioAPI_SC_USE_KEY_B(0x61)의 값을 가질 수 있다.

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

KeyValue:
사용할 6bytes 길이의 키 값.

NewAccessBit:
새로운 Access Bits를 담고 있는 4bytes 길이의 버퍼 포인터.

NewKeyA:
새로운 Key A를 담고 있는 6bytes 길이의 버퍼 포인터.

NewKeyB:
새로운 Key B를 담고 있는 6bytes 길이의 버퍼 포인터.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_ReqA**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_ReqA (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [OUT] UCBioAPI_UINT8*     pResultBuff,
              [OUT] UCBioAPI_UINT16*    nResultLen,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 ATQ를 얻는다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_WupA**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_WupA (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [OUT] UCBioAPI_UINT8*     pResultBuff,
              [OUT] UCBioAPI_UINT16*    nResultLen,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 ATQ를 얻는다.
UCBioAPI_SC_ReqA 함수와 동일하지만 Halt 상태인 카드도 응답하는 점이 다르다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_Select**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_Select (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [OUT] UCBioAPI_UINT8*     pResultBuff,
              [OUT] UCBioAPI_UINT16*    nResultLen,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

Type A 카드로부터 ATQ를 얻은 상태에서, 5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 카드의 SAK 값과 UID 값을 얻어온다.
SAK는 1byte의 길이 값을 가지고 UID는 가변 길이의 길이 값을 가진다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pResultBuff:
결과값이 담길 버퍼 포인터.
SAK(1byte) + UID 가 순서대로 담겨져 있다.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_HaltA**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_HaltA (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

Select 상태인 Type A 카드를 Halt 상태로 바꾼다.
이렇게 Halt 상태가 된 카드는 UCBioAPI_SC_ReqA 함수에 대해서는 응답을 하지 않고 UCBioAPI_SC_WupA 함수에 대해서만 응답하게 된다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_Rats**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_Rats (
                [IN]  UCBioAPI_HANDLE     hHandle,
                [IN]  UCBioAPI_UINT8      fsdi,
                [IN]  UCBioAPI_UINT8      cid,
                [OUT] UCBioAPI_UINT8*     pResultBuff,
                [OUT] UCBioAPI_UINT16*    nResultLen,
                [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 카드의 ATS(Answer To Select) 값을 얻어온다. ATS를 얻기 위해서는 SAK 값이 0x2X(ISO14443-4 지원)일 때 가능하다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

Fsdi:
FSDI(Frame Size for proximity coupling Device Integer)의 설정을 통해 PCD가 수신 할 수 있는 프레임의 최대 크기를 정할 수 있다. (0x00 ~ 0x0F까지의 값)

cid:
CID(Card Identifier)를 통해 각 개별 카드를 선택적으로 호출하는 것이 가능하다. (0x00 ~ 0x0F까지의 값)

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE

UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_PpsRequest**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_PpsRequest (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [IN]  UCBioAPI_UINT8      cid,
              [IN]  UCBioAPI_UINT8      pps0,
              [IN]  UCBioAPI_UINT8      pps1,
              [OUT] UCBioAPI_UINT8*     pResultBuff,
              [OUT] UCBioAPI_UINT16*    nResultLen,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 카드의 PPSS 값을 얻어온다.
ATS의 값을 통해 변화될 수 있는 Parameter를 지원한다면, PCD에 의해 사용될 수 있다. 즉, ATS 내의 선택적 Parameter인 DS와 DR에서 더 높은 보레이트를 지원하면 양방향으로의 보레이트는 서로 독립적으로 2,4,8배로 증가할 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

cid:
Rats 호출 시 선택한 CID(Card Identifier) 값. (0x00 ~ 0x0F까지의 값)

pps0:
PPS1을 전송하는지 하지 않는지를 나타낸다.
0x11이면 전송, 0x01이면 전송하지 않음을 의미.

pps1:
상위 2bytes는 DRI이며 하위 2bytes는 DS를 의미.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.

UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_BlockFormat**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_BlockFormat (
              [IN]  UCBioAPI_HANDLE    hHandle,
              [IN]  UCBioAPI_UINT8     pcb,
              [IN]  UCBioAPI_UINT8     CidOrNad,
              [IN]  UCBioAPI_UINT8*    inf,
              [IN]  UCBioAPI_UINT8     infLen,
              [OUT] UCBioAPI_UINT8*    pResultBuff,
              [OUT] UCBioAPI_UINT16*   nResultLen,
              [IN]  UCBioAPI_UINT16    wLed);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 데이터를 읽어온다.
데이터 계층에서의 블록 교환에 관한 명령(T=1).
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pcb:
PCB(Protocol Control Byte)를 의미. (필수)
데이터 전송을 제어하는 데 필요한 정보를 가지고 있음. I – Block, R – Block, S – Block으로 나뉘어짐. 데이터를 읽기 위해서 보통 I – Block을 사용하며 PCB 값은 0x0A, 0x0B, 0x0A 식으로 Bit0을 Toggle하여 보내야 함.

CidOrNad:
Rats로 PICC에 지정한 CID 또는 NAD 값. (선택)

inf:
INF(Information Field)를 의미.
해당 파일을 읽으려면 특별한 포맷을 알고 있어야 함.

infLen:
inf의 길이 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:
UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_Deselect**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_Deselect (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [IN]  UCBioAPI_UINT8      CidOrNad,
              [OUT] UCBioAPI_UINT8*     pResultBuff,
              [OUT] UCBioAPI_UINT16*    nResultLen,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 Active 상태인 Type A카드가 있으면 카드를 비활성화 시킨다.
Active State인 카드를 Deselect하면, 카드는 RF영역을 벗어나지 않는 한 BlockFormat에 대한 응답을 하지 않는다. 반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

CidOrNad:
Rats로 PICC에 지정한 CID 또는 NAD 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

■ **UCBioAPI_SC_TypeA_ActiveState**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_TypeA_ActiveState (
              [IN]  UCBioAPI_HANDLE     hHandle,
              [OUT] UCBioAPI_UINT8*     pResultBuff,
              [OUT] UCBioAPI_UINT16*    nResultLen,
              [IN]  UCBioAPI_UINT16     wLed);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드를 ReqA, Select, Rats까지 한꺼번에 수행해 카드로부터 ATS를 얻어온다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

hHandle:
UCBioAPI SDK의 Handle 값.

pResultBuff:
결과값이 담길 버퍼 포인터.

nResultLen:
결과값의 길이.

wLed:
성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
UCBioAPI_SC_LED_TOGGLE(1)을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌게 된다.

Returns:

UCBioAPIERROR_NONE
UCBioAPIERROR_INVALID_HANDLE
UCBioAPIERROR_FUNCTION_NOT_SUPPORTED
UCBioAPIERROR_DEVICE_NOT_OPENED
UCBioAPIERROR_SC_FUNCTION_FAILED
UCBioAPIERROR_SC_NOT_SUPPORTED_DEVICE
UCBioAPIERROR_SC_NOT_SUPPORTED_FIRMWARE

6.API Reference for COM

이 장에서는 COM 모듈인 UCBioBSPCOM.dll을 사용하기 위한 Property 및 Method들에 대해 설명한다.

6.1. UCBioBSP Object

UCBioBSPCOM.dll을 사용하기 위한 Main Object로서 UCBioBSP SDK의 버전 정보 및 기본 설정 등의 기능을 제공하고 각종 기능별 Child Object를 얻어오는 기본 Object로 동작한다. 때문에 SDK를 사용하기 위해서는 이 Object가 반드시 선언되어야 한다.

6.1.1. Methods

UCBioBSP Object의 각종 Method에 대해 설명한다.

■ **SetSkinResource**

Prototype:

```
HRESULT SetSkinResource(BSTR bszSkinPath);
```

Description:

UCBioAPI SDK의 User Interface를 위한 Skin을 변경한다.
UCBioAPI SDK에서는 사용하는 등록 및 인증용 화면을 Skin 방식의 UI로 사용하고 있다. 때문에 UCBioBSP SDK에서 기본 제공하는 UI를 사용하지 않고 다른 언어로 된 UI나 또는 다른 형태의 UI를 사용하고자 하는 경우에는 사용자 정의 Skin을 만들어 사용 할 수 있다.

Parameters:

bszSkinPath:
바꾸고자 하는 Skin 파일의 Full path.

Related properties:

ErrorCode, ErrorDescription

■ **SetCaptureQuality**

Prototype:

```
HRESULT SetCaptureQuality(
                          long nScanQuality,
                          long nEnrollQuality,
                          long nVerifyQuality,
                          long nIdentifyQuality);
```

Description:
UCBioAPI SDK의 입력 지문에 대한 품질 값을 설정한다. 지문 입력시 입력된 지문의 품질이 해당 설정값보다 이상인 경우만 입력지문으로 허용된다. 설정 가능값은 1부터100이며 기본값은 50이다. 설정값이 0인 경우 해당 설정값은 적용되지 않고 이전값이 유지된다.

Parameters:
nScanQuality : 입력된 지문의 순수한 이미지에 대한 품질의 최소 설정값
nEnrollQuality : 추출된 특징점 품질값에 대한 등록시 적용하는 최소 설정값
nVerifyQuality : 추출된 특징점 품질값에 대한 1:1매칭시 적용하는 최소 설정값
nIdentifyQuality : 추출된 특징점 품질값에 대한 1:N 매칭시 적용하는 최소 설정값
*** 현재 nScanQuality, nEnrollQuality 만 적용됨.**

Returns: UCBioAPIERROR_TRUE

6.1.2. Properties

UCBioBSP Object의 각종 Property에 대해 설명한다.

■ **ErrorCode**

Prototype:

```
[ReadOnly]  long          ErrorCode;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.
Child Object의 Method 및 Property 설정 중 발생한 오류도 이 값으로 얻을 수 있다.

■ **ErrorDescription**

Prototype:

```
[ReadOnly]  BSTR          ErrorDescription;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 문자열로 담겨있다.
ErrorCode에 대한 오류 값을 문자열로 출력하고자 할 때 사용할 수 있다.
ErrorCode와 마찬가지로 Child Object의 Method 및 Property 설정 중 발생한 오류도 이 값으로 얻을 수 있다.

■ **Device**

Prototype:

```
[ReadOnly]  VARIANT       Device;
```

Description:

디바이스 관련 명령 집합을 가지고 있는 개체의 인터페이스를 얻는다. 디바이스의 사용 시작 및 종료, 디바이스 선택, 옵션 값 설정 등의 기능을 수행하기 위해 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IDevice 설명 참조.

■ **Extraction**

Prototype:

```
[ReadOnly]  VARIANT       Extraction;
```

Description:

지문 이미지를 얻어와 특징점을 추출하는 기능과 관련된 인터페이스를 얻는다. 지문 획득 및 등록과 관련한 기능을 수행하기 위해 이 인터페이스를 얻어와 사용해야 한다. 자세한

것은 IExtraction 설명 참조.

■ **Matching**

Prototype:

```
[ReadOnly]  VARIANT      Matching;
```

Description:

지문 데이터를 이용해 인증을 수행하는 기능과 관련된 인터페이스를 얻는다. 지문 인증과 관련한 기능을 수행하기 위해 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IMatching 설명 참조.

■ **FPData**

Prototype:

```
[ReadOnly]  VARIANT      FPData;
```

Description:

지문 데이터를 얻어오거나 변환하는 기능과 관련된 인터페이스를 얻는다. 지문 데이터를 다른 형태로 변환 하고자 할 경우 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IFPData 설명 참조.

■ **FPImage**

Prototype:

```
[ReadOnly]  VARIANT      FPImage;
```

Description:

지문 데이터를 이미지로 얻어오거나 이미지 파일로 저장하는 기능과 관련된 인터페이스를 얻는다. 지문 데이터를 이미지로 변환 하고자 할 경우 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IFPImage 설명 참조.

■ **FastSearch**

Prototype:

```
[ReadOnly]  VARIANT      FastSearch;
```

Description:

1:N 고속 인증을 수행하기 위한 FastSearch 기능과 관련된 인터페이스를 얻는다. FastSearch 관련 DB 관리 및 인증 수행 등과 같은 기능을 사용하고자 할 경우 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 IFastSearch 설명 참조.

■ **SmartCard**

Prototype:

```
[ReadOnly]  VARIANT       SmartCard;
```

Description:

스마트카드와 관련된 인터페이스를 얻는다. 스마트카드에 데이터를 저장 및 읽어오는 기능 등을 사용하기 위해 이 인터페이스를 얻어와 사용해야 한다. 자세한 것은 ISmartCard 설명 참조.

■ **CheckValidityModule**

Prototype:

```
[ReadOnly]  BOOL          CheckValidityModule;
```

Description:

UCBioBSP SDK의 유효성을 검사해 그 값을 가진다.
이 값이 TRUE이면 유효성에 문제가 없다. 하지만 만약 이 값이 FALSE이라면 SDK의 Core 모듈인 UCBioBSP.dll 파일이 임의로 변경되었거나 수정 되었을 가능성이 있으므로 확인을 요한다.

■ **MajorVersion**

Prototype:

```
[ReadOnly]  BSTR          MajorVersion;
```

Description:

UCBioBSP SDK의 Major1, Major2 버전 값을 문자열로 가진다. 예를 들어 SDK의 버전이 v3.4.0.2인 경우 Major Version에는 "3.4"이, Minor Version에는 "0.2"이라는 값이 들어간다.

■ **MinorVersion**

Prototype:

```
[ReadOnly]  BSTR          MinorVersion;
```

Description:

UCBioBSP SDK의 Minor1, Minor2 버전 값을 문자열로 가진다. 예를 들어 SDK의 버전이 v3.4.0.2인 경우 Major Version에는 "3.4"이, Minor Version에는 "0.2"이라는 값이 들어간다.

■ **BuildNumber**

Prototype:

```
[ReadOnly]  BSTR          BuildNumber;
```

Description: 더 이상 지원하지 않음

■ **MaxFingersForEnroll**

Prototype:

```
[Read/Write] long        MaxFingersForEnroll;
```

Description:

지문 등록을 할 경우 최대 등록 가능하게 할 손가락의 개수를 지정한다.
예를 들어 이 값이 2로 지정될 경우 IExtraction의 Enroll Method를 이용해 지문을 등록 할 경우 최대 2개의 손가락만 등록이 가능하다.
디폴트로 이 값은 10을 가진다.

Related methods:

IExtraction.Enroll

■ **NecessaryEnrollNum**

Prototype:

```
[Read/Write] long        NecessaryEnrollNum;
```

Description:

지문 등록을 할 경우 최소 등록해야 할 손가락의 개수를 지정한다.
이 값은 반드시 MaxFingersForEnroll 값보다는 작거나 같아야 한다.
예를 들어 이 값이 2로 지정될 경우 IExtraction의 Enroll Method를 이용해 지문을 등록 할 경우 최소 2개의 손가락이 등록되어야지만 등록 과정의 종료가 가능하다.
디폴트로 이 값은 1을 가진다.

Related methods:

IExtraction.Enroll

■ **SamplesPerFinger**

Prototype:

```
[Read/Write] long        SamplesPerFinger;
```

Description:

지문 등록 시 손가락당 몇 개의 Sample이 저장되는지를 지정한다. 현재는 2로 고정되어 있으며 수정 할 수 없다.

■ **DefaultTimeout**

Prototype:

```
[Read/Write] long        DefaultTimeout;
```

Description:

지문 인증 및 등록 시 지문을 획득하기 위해 디바이스가 동작하는 기본 최대 시간을 ms 단위로 지정한다. Timeout은 향후 함수 호출 시 따로 지정 할 수 있으나 이때 -1을 지정한 경우 이 값이 사용된다.
디폴트로 이 값은 10000(10초)을 가진다.

Related methods:

IExtraction.Enroll, IExtraction.Capture, IMatching.Verify

■ **SecurityLevelForEnroll / SecurityLevelForVerify / SecurityLevelForIdentify**

Prototype:

```
[Read/Write] long        SecurityLevelForEnroll;
[Read/Write] long        SecurityLevelForVerify;
[Read/Write] long        SecurityLevelForIdentify;
```

Description:

지문 등록 / 인증 / 1:N 인증을 위해 사용되어질 인증 보안 레벨을 각각에 대해 설정 할 수 있다. 이 값은 아래와 같은 값을 가질 수 있다.

```
1 -       LOWEST
2 -       LOWER
3 -       LOW
4 -       BELOW_NORMAL
5 -       NORMAL
6 -       ABOVE_NORMAL
7 -       HIGH
8 -       HIGHER
9 -       HIGHEST
```

기본값으로 Enroll/Verify는 5의 값을 가지고 Identify는 6의 값을 가진다.

Related methods:

IExtraction.Enroll, IMatching.Verify, IMatching.VerifyMatch

■ **WindowStyle**

Prototype:

```
[Read/Write] long        WindowStyle;
```

Description:
윈도우가 화면에 표시되는 형태를 지정 할 수 있다. 윈도우가 팝업 형태로 뜰 것인지 아니면 다른 윈도우의 영역에 지문만 표시 할 것인지를 지정 할 수 있다.

```
0 -        POPUP
1 -        INVISIBLE
```

POPUP으로 지정 할 경우 일반적으로 새로운 윈도우가 떠서 지문 등록 및 인증을 수행한다. 하지만 IExtraction의 Capture와 IMatching의 Verify Method의 경우 INVISIBLE 값을 지정 할 수 있는데 그렇게 할 경우 화면에 UI를 표시 하지 않으면서 지문 입력이나 인증을 수행 할 수 있다. 또한 이렇게 INVISIBLE로 지정 할 경우 FingerWnd Property의 값이 Null이 아니라면 그 윈도우에 지문 이미지를 표시하도록 할 수도 있다.

Related methods:
IExtraction.Enroll, IExtraction.Capture, IMatching.Verify, IDevice.Adjust

■ **WindowOption**

Prototype:

```
[Read/Write] BOOL        WindowOption(long Option);
```

Description:
윈도우가 화면에 표시되는 형태에 대한 Flag를 지정 할 수 있다. 지문이 화면에 표시되지 않게 하거나 지문 등록 시 Welcome 페이지를 없애는 등의 설정을 할 수 있다. 이 세가지 Flag는 중복해서 지정 가능하다. Option값에 아래와 같은 값을 지정하고 TRUE/FALSE 값을 지정하거나 값을 얻어오면 된다.

```
0x00010000 – NO_FPIMG
0x00020000 – NO_WELCOME
0x00040000 – NO_TOPMOST
```

0x00010000 - NO_FPIMG:
보안상 지문 이미지를 화면에 표시하고자 하지 않을 경우 이 Style을 지정한다.
0x00020000 - NO_WELCOME:
지문 등록 UI에서 첫 번째 Page인 Welcome Page를 표시하지 않는다.
0x00040000 - NO_TOPMOST:
현재는 UI가 모두 최상위 윈도우로 뜨게 되어있으나 이것을 일반 윈도우로 뜨게 할 경우 이 Style을 지정한다.

Related methods:
IExtraction.Enroll, IExtraction.Capture, IMatching.Verify, IDevice.Adjust

■ **ParentWnd**

Prototype:

```
[Read/Write] long        ParentWnd;
```

Description:
지문 등록 및 인증 윈도우가 뜰 때 그 기반이 되는 부모 윈도우의 Handle을 지정. 기본값은 Null 값을 가진다.

Related methods:
IExtraction.Enroll, IExtraction.Capture, IMatching.Verify, IDevice.Adjust

■ **FingerWnd**

Prototype:

```
[Read/Write] long        FingerWnd;
```

Description:
WindowStyle이 INVISIBLE(1) 값일 경우 지문 이미지를 그려줄 특정 윈도우의 Handle을 지정. 기본값은 Null 값을 가진다.

Related methods:
IExtraction.Capture, IMatching.Verify

■ **CaptionMsg**

Prototype:

```
[Read/Write] BSTR        CaptionMsg;
```

Description:
지문 등록 시 사용자가 취소를 선택 할 경우 취소 메시지를 보여줄 윈도우의 Caption에 표시할 메시지를 지정.

Related methods:
IExtraction.Enroll

■ **CancelMsg**

Prototype:

```
[Read/Write] BSTR        CancelMsg;
```

Description:

지문 등록 시 사용자가 취소를 선택 할 경우 보여줄 취소 메시지를 지정.

Related methods:

IExtraction.Enroll

■ **FPForeColor / FPBackColor**

Prototype:

```
[Read/Write] BSTR        FPForeColor;
[Read/Write] BSTR        FPBackColor;
```

Description:

화면에 표시할 지문 이미지의 지문 색상과 배경 값을 문자열로 된 RGB 값으로 지정.
"FFFFFF" 과 같이 들어있다면 각 2byte가 HEX값으로 된 RGB 값이라 보면 된다.

Related methods:

IExtraction.Enroll, IExtraction.Capture, IMatching.Verify

■ **DisableFingerForEnroll**

Prototype:

```
[Read/Write] BOOL        DisableFingerForEnroll(long nFingerID);
```

Description:

지문 등록 시 등록하지 못하게 할 손가락을 지정 할 수 있다. 총 10개의 값을 담을 수 있는 배열 값으로 각 손가락 값에 따른 등록 유무를 0 또는 1로 지정하며 된다.
1번 Index부터 오른손 엄지, 검지, 중지, 가락지, 약지 순으로 증가하며 6번 Index부터 왼손 엄지, 검지, 중지, 가락지, 약지 순으로 증가한다. 0번 Index는 사용하지 않는다.
예를 들어 왼손 엄지를 등록하지 못하게 하려면 다음과 같이 지정하면 된다.

```
objUCBioBSP.DisableFingerForEnroll(6) = True;
```

단, 지문 수정 모드일 경우 사용 금지를 한 손가락이라도 이미 등록된 손가락이라면 등록 상태를 표시해 준다.

Related methods:

IExtraction.Enroll

6.2. IDevice Interface

디바이스 관련 명령 집합을 가지고 있는 개체의 인터페이스. 디바이스의 사용 시작 및 종료, 디바이스 선택, 옵션 값 설정 등의 기능을 수행하기 위해 이 인터페이스를 얻어와 사용해야 한다.

6.2.1. Methods

IDevice Interface의 각종 Method에 대해 설명한다.

■ **Open**

Prototype:

```
HRESULT Open(long nDeviceID);
```

Description:

원하는 디바이스를 열고 초기화 한다.

Parameters:

nDeviceID:

열고자 하는 디바이스 ID 값. UCBioAPI_DEVICE_ID 참조.

Related properties:

ErrorCode, ErrorDescription

OpenedDeviceID, ImageWidth, ImageHeight, Brightness, Contrast, Gain

■ **Close**

Prototype:

```
HRESULT Close(long nDeviceID);
```

Description:

열려진 디바이스를 닫고 사용을 종료한다.

Parameters:

nDeviceID:

닫고자 하는 디바이스 ID 값. UCBioAPI_DEVICE_ID 참조.

Related properties:

ErrorCode, ErrorDescription

■ **Enumerate**

Prototype:

```
HRESULT Enumerate();
```

Description:

현재 시스템에 설치된 모든 디바이스를 검색해 목록을 만든다. 만들어진 목록은 관련 Property를 통해 얻을 수 있다.

Related properties:

ErrorCode, ErrorDescription
EnumCount, EnumDeviceID, EnumDeviceNameID, EnumDeviceInstance, EnumDeviceName, EnumDeviceDescription, EnumDeviceDll, EnumDeviceSys, EnumDeviceAutoOn, EnumDeviceBrightness, EnumDeviceContrast, EnumDeviceGain

■ **Adjust**

Prototype:

```
HRESULT Adjust();
```

Description:

현재 열려진 디바이스의 밝기를 조정하는 UI를 띄운다. 하지만 현재 지원하는 디바이스는 내부적으로 자동 밝기 조절과정을 거치므로 이 함수를 사용 할 수 없다.

Related properties:

ErrorCode, ErrorDescription

6.2.2. Properties

IDevice Interface의 각종 Property에 대해 설명한다.

■ **ErrorCode**

Prototype:

```
[ReadOnly]  long          ErrorCode;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.

■ **ErrorDescription**

Prototype:

```
[ReadOnly]  BSTR          ErrorDescription;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 문자열로 담겨있다.
ErrorCode에 대한 오류 값을 문자열로 출력하고자 할 때 사용할 수 있다.

■ **EnumCount**

Prototype:

```
[ReadOnly]  long          EnumCount;
```

Description:

IDevice.Enumerate Method를 호출 후 검색되어진 디바이스의 개수가 담겨있다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:

IDevice.Enumerate

■ **EnumDeviceID**

Prototype:

```
[ReadOnly]  long          EnumDeviceID(long nIndex);
```

Description:

IDevice.Enumerate Method를 호출 후 검색되어진 디바이스 ID의 목록이 담겨있다.
nIndex는 0부터 (EnumCount -1)의 값을 가질 수 있다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:
IDevice.Enumerate

■ **EnumDeviceNameID**

Prototype:

```
[ReadOnly]  long          EnumDeviceNameID(long nIndex);
```

Description:
IDevice.Enumerate Method를 호출 후 검색되어진 디바이스 이름 ID의 목록이 담겨있다.
nIndex는 0부터 (EnumCount -1)의 값을 가질 수 있다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:
IDevice.Enumerate

■ **EnumInstance**

Prototype:

```
[ReadOnly]  long          EnumDeviceInstance(long nIndex);
```

Description:
IDevice.Enumerate Method를 호출 후 검색되어진 디바이스 Instance의 목록이 담겨있다.
nIndex는 0부터 (EnumCount -1)의 값을 가질 수 있다.
현재는 디바이스별 Instance를 지원하지 않으므로 사용하지 않는다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:
IDevice.Enumerate

■ **EnumDeviceName**

Prototype:

```
[ReadOnly]  BSTR          EnumDeviceName(long nIndex);
```

Description:
IDevice.Enumerate Method를 호출 후 검색되어진 디바이스 이름의 목록이 담겨있다.
nIndex는 0부터 (EnumCount -1)의 값을 가질 수 있다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:
IDevice.Enumerate

■ **EnumDeviceDescription**

Prototype:

```
[ReadOnly]  BSTR        EnumDeviceDescription(long nIndex);
```

Description:
IDevice.Enumerate Method를 호출 후 검색되어진 디바이스 설명에 대한 목록이 담겨있다.
nIndex는 0부터 (EnumCount -1)의 값을 가질 수 있다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:
IDevice.Enumerate

■ **EnumDeviceDll / EnumDeviceSys**

Prototype:

```
[ReadOnly]  BSTR        EnumDeviceDll(long nIndex);
[ReadOnly]  BSTR        EnumDeviceSys(long nIndex);
```

Description:
IDevice.Enumerate Method를 호출 후 검색되어진 디바이스 Dll명과 Sys명에 대한 목록이 담겨있다.
nIndex는 0부터 (EnumCount -1)의 값을 가질 수 있다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:
IDevice.Enumerate

■ **EnumDeviceAutoOn**

Prototype:

```
[ReadOnly]  long        EnumDeviceAutoOn(long nIndex);
```

Description:
IDevice.Enumerate Method를 호출 후 검색되어진 디바이스가 AutoOn을 지원하는지에 대한 값의 목록이 담겨있다. 현재 AutoOn에 대한 값은 지원하지 않는다.
nIndex는 0부터 (EnumCount -1)의 값을 가질 수 있다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:
IDevice.Enumerate

■ **EnumDeviceBrightness / EnumDeviceContrast / EnumDeviceGain**

Prototype:

```
[ReadOnly]  long        EnumDeviceBrightness(long nIndex);
[ReadOnly]  long        EnumDeviceContrast(long nIndex);
[ReadOnly]  long        EnumDeviceGain(long nIndex);
```

Description:
IDevice.Enumerate Method를 호출 후 검색되어진 디바이스의 밝기 / 대비 / Gain 값의 목록이 담겨있다.
nIndex는 0부터 (EnumCount -1)의 값을 가질 수 있다.
반드시 IDevice.Enumerate 호출 후 사용하여야 한다.

Related methods:
IDevice.Enumerate

■ **OpenedDeviceID**

Prototype:

```
[ReadOnly]  long        OpendedDeviceID;
```

Description:
현재 열려진 디바이스 ID 값을 담고 있다.

■ **DeviceNameID / DeviceInstance**

Prototype:

```
[ReadOnly]  long        DeviceNameID(long nDeviceID);
[ReadOnly]  long        DeviceInstance(long nDeviceID);
```

Description:
주어진 nDeviceID로부터 디바이스 이름 ID와 디바이스 Instance를 구한다.
각각의 값은 nDeviceID의 하위 2bytes와 상위 2bytes를 의미한다.
현재는 디바이스별 Instance를 지원하지 않으므로 사용하지 않는다.

■ **DeviceID**

Prototype:

```
[ReadOnly]  long          DeviceID(long nNameID, long nInstance);
```

Description:

주어진 nNameID와 nInstance를 이용해 디바이스 ID 값을 구한다.
현재는 디바이스별 Instance를 지원하지 않으므로 사용하지 않는다.

■ **ImageWidth / ImageHeight**

Prototype:

```
[ReadOnly]  long          ImageWidth(long nDeviceID);
[ReadOnly]  long          ImageHeight(long nDeviceID);
```

Description:

주어진 nDeviceID를 이용해 그 디바이스의 이미지 크기를 얻는다.
nDeviceID는 현재 열려진 디바이스 ID 값과 같아야 하며 반드시 디바이스가 Open되 상태에서 사용하여야 한다.

Related methods:

IDevice.Open, IDevice.OpenedDeviceID

■ **Brightness / Contrast / Gain**

Prototype:

```
[Read/Write] long         Brightness(long nDeviceID);
[Read/Write] long         Contrast(long nDeviceID);
[Read/Write] long         Gain(long nDeviceID);
```

Description:

주어진 nDeviceID를 이용해 그 디바이스의 밝기 / 대비 / Gain 값을 얻거나 지정한다.
nDeviceID는 현재 열려진 디바이스 ID 값과 같아야 하며 반드시 디바이스가 Open되 상태에서 사용하여야 한다.

Related methods:

IDevice.Open, IDevice.OpenedDeviceID

■ **IsFingerExisted**

Prototype:

```
[ReadOnly]  BOOL          IsFingerExisted;
```

Description:
현재 열려진 디바이스에 손가락이 올려져 있는지를 검사해 그 결과를 담고 있다.
이 Property를 호출하는 순간에 검사를 진행하게 된다.
반드시 디바이스가 Open되 상태에서 사용하여야 한다.

Related methods:
IDevice.Open

6.3. IExtraction Interface

지문 이미지를 얻어와 특징점을 추출하는 기능과 관련된 인터페이스. 지문 획득 및 등록과 관련한 기능을 수행하기 위해 이 인터페이스를 얻어와 사용해야 한다.

6.3.1. Methods

IExtraction Interface의 각종 Method에 대해 설명한다.

■ **Auto Detect**

Prototype:

```
HRESULT SetAutoDetect(long nFlag);
```

Description:

지문 입력을 자동으로 감지하여 지문을 입력 받는다

Parameters:

nFlag:

자동 감지 기능 설정 유무

```
0 -       Not Use
1 -       Use Autodetect
```

Related properties:

Capture, Enroll

FIR, FIRLength, TextFIR

Related methods:

IDevice.Open

■ **Capture**

Prototype:

```
HRESULT Capture([in, optional]long nPurpose);
```

Description:

현재 열려진 디바이스로부터 1개의 지문을 획득하여 FIR Handle을 만든다. 이렇게 획득된 데이터는 FIR이나 TextFIR Property를 이용해 얻을 수 있다. 이 Method는 디바이스를 사용하므로 사용 전 반드시 디바이스가 Open 되어 있어야 한다.

Parameters:

nPurpose:

FIR 데이터의 Purpose 값을 지정 할 수 있다. 가질 수 있는 값은 다음과 같다.

```
1 -       VERIFY
2 -       IDENTIFY
3 -       ENROLL
4 -       ENROLL_FOR_VERIFICATION_ONLY
5 -       ENROOL_FOR_IDENTIFICATION_ONLY
6 -       AUDIT
16 -      UPDATE
```

각각의 값들은 FIR 데이터에 대한 참조용 값으로만 사용되고 인증 시에는 영향을 미치지 않는다. 만약 이 값을 ENROLL 관련 Purpose로 지정 할 경우 Enroll Method와 동일하게 동작한다.
좀 더 자세한 설명은 UCBioAPI_FIR_PURPOSE 정의 참조.

Related properties:
ErrorCode, ErrorDescription
FIR, FIRLength, TextFIR

Related methods:
IDevice.Open

■ **Enroll**

Prototype:

```
HRESULT Enroll(VARIANT payload, [in, optional]VARIANT storedFIR);
```

Description:
현재 열려진 디바이스로부터 1명의 지문을 획득하여 FIR 데이터를 만든다. 1명의 지문이기 때문에 다수개의 손가락에 대한 지문을 한번에 등록 받아 하나의 FIR 데이터로 만들게 된다. 이렇게 등록된 지문은 FIR이나 TextFIR Property를 이용해 얻을 수 있다. 이 Method는 디바이스를 사용하므로 사용 전 반드시 디바이스가 Open 되어 있어야 한다.

Parameters:
payload:
만들어질 FIR 데이터에 담겨질 Payload 데이터. 문자열 또는 Binary 데이터일 수 있다.
이 값은 Null이 사용 되어 질 수 있고 Null이 지정되면 기존 데이터의 Payload 값이 유지된다.

storedFIR:
기존 FIR 데이터를 수정 할 경우 이 값을 지정한다. 이 값을 지정하지 않을 경우 새로운

지문을 입력 받게 된다.

Related properties:
ErrorCode, ErrorDescription
FIR, FIRLength, TextFIR

Related methods:
IDevice.Open

6.3.2. Properties

IExtraction Interface의 각종 Property에 대해 설명한다.

■ **ErrorCode**

Prototype:

```
[ReadOnly]  long          ErrorCode;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.

■ **ErrorDescription**

Prototype:

```
[ReadOnly]  BSTR          ErrorDescription;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 문자열로 담겨있다.
ErrorCode에 대한 오류 값을 문자열로 출력하고자 할 때 사용할 수 있다.

■ **FIR**

Prototype:

```
[ReadOnly]  VARIANT       FIR;
```

Description:

Capture나 Enroll Method 호출 후 획득된 FIR 데이터를 Binary stream 데이터로 얻어온다.
데이터의 길이 값은 FIRLength Property에 담겨있다.
얻어오기 위한 FIR Format은 미리 FIRFormat property에 지정해 두어야 한다.

Related methods:

Capture, Enroll

Related properties:

FIRLength, FIRFormat

■ **FIRLength**

Prototype:

```
[ReadOnly]  long          FIRLength;
```

Description:
Capture나 Enroll Method 호출 후 획득된 Binary stream FIR 데이터의 길이 값을 얻어온다.

Related methods:
Capture, Enroll

Related properties:
FIR

■ **TextFIR**

Prototype:

```
[ReadOnly]  BSTR          TextFIR;
```

Description:
Capture나 Enroll Method 호출 후 획득된 FIR 데이터를 Text string 형태의 문자열로 얻어온다. 문자열이므로 길이 값은 따로 지정하지 않는다.
얻어오기 위한 FIR Format은 미리 FIRFormat property에 지정해 두어야 한다.

Related methods:
Capture, Enroll

Related properties:
FIRFormat

■ **FIRFormat**

Prototype:

```
[Read/Write] long         FIRFormat;
```

Description:
얻어올 FIR의 데이터 포맷을 지정한다. 향후 FIR 데이터의 구조가 바뀔 경우 이 포맷값이 바뀌어 하위 호환성을 유지 할 수 있다. 현재는 STANDARD(1) 포맷만 지원하고 있다.

Related properties:
FIR, TextFIR

6.4. IMatching Interface

지문 데이터를 이용해 인증을 수행하는 기능과 관련된 인터페이스. 지문 인증과 관련한 기능을 수행하기 위해 이 인터페이스를 얻어와 사용해야 한다.

6.4.1. Methods

IMatching Interface의 각종 Method에 대해 설명한다.

■ **VerifyMatch**

Prototype:

```
HRESULT VerifyMatch (VARIANT processedFIR, VARIANT storedFIR);
```

Description:

기존에 획득된 두 개의 FIR 데이터를 서로 비교하여 그 인증 결과를 얻어온다. 인증이 성공 할 경우 등록용 FIR 데이터에 들어있던 Payload 값을 얻어 올 수도 있다.

Parameters:

processedFIR:

인증을 요하는 FIR 데이터. Binary stream형태의 FIR 데이터일수도 있고 Text string형태의 TextFIR 데이터일수도 있다. 어떤 것을 사용하건 상관없다.

storedFIR:

인증을 요하는 등록용 FIR 데이터. Binary stream형태의 FIR 데이터일수도 있고 Text string 형태의 TextFIR 데이터일수도 있다. 어떤 것을 사용하건 상관없다.

Related properties:

ErrorCode, ErrorDescription
MatchingResult, IsPayloadExisted, Payload, PayloadLength, TextPayload

■ **Verify**

Prototype:

```
HRESULT Verify (VARIANT storedFIR);
```

Description:

기존에 획득된 FIR 데이터와 현재 디바이스로부터 라이브로 입력 받는 지문 데이터를 서로 비교하여 그 인증 결과를 얻어온다. 인증이 성공 할 경우 등록용 FIR 데이터에 들어있던 Payload 값을 얻어 올 수도 있다. VerifyMatch Method와 거의 동일하지만 라이브 지문을 직접 입력 받아 인증을 수행하는 점이 다르다.
즉, 내부적으로 IExtraction.Capture Method와 IMathing.VerifyMatch Method가 같이 수행되는 것으로 생각 할 수 있다. 이 함수는 디바이스를 사용하므로 사용 전 반드시 디바이스가

Open 되어 있어야 한다.

Parameters:
storedFIR:
인증을 요하는 등록용 FIR 데이터. Binary stream형태의 FIR 데이터일수도 있고 Text string 형태의 TextFIR 데이터일수도 있다. 어떤 것을 사용하건 상관없다.

Related properties:
ErrorCode, ErrorDescription
MatchingResult, IsPayloadExisted, Payload, PayloadLength, TextPayload

Related methods:
IDevice.Open

6.4.2. Properties

IMatching Interface의 각종 Property에 대해 설명한다.

■ **ErrorCode**

Prototype:

```
[ReadOnly]  long          ErrorCode;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.

■ **ErrorDescription**

Prototype:

```
[ReadOnly]  BSTR          ErrorDescription;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 문자열로 담겨있다.
ErrorCode에 대한 오류 값을 문자열로 출력하고자 할 때 사용할 수 있다.

■ **MatchingResult**

Prototype:

```
[ReadOnly]  BOOL          MatchingResult;
```

Description:

Verify나 VerifyMatch Method 호출 후 그 인증에 대한 결과값이 담겨 있다.

Related methods:

Verify, VerifyMatch

■ **IsPayloadExisted**

Prototype:

```
[ReadOnly]  BOOL          IsPayloadExisted;
```

Description:

Verify나 VerifyMatch Method 호출 후 인증이 성공한 경우 FIR 내부에 담겨있던 Payload를 얻어 올 수 있는데, Payload 값이 존재하는지 여부에 대한 값을 담고 있다.

Related methods:

Verify, VerifyMatch

Related properties:
Payload, PayloadLength, TextPayload

■ **Payload**

Prototype:

```
[ReadOnly]  VARIANT       Payload;
```

Description:
Verify나 VerifyMatch Method 호출 후 인증이 성공한 경우 FIR 내부에 담겨있던 Payload를 얻어 올 수 있는데, 그 Payload 값이 Binary stream 형태로 담겨있다.
Binary stream 데이터의 길이 값은 PayloadLength property에 담겨있다.

Related methods:
Verify, VerifyMatch

Related properties:
PayloadLength

■ **PayloadLength**

Prototype:

```
[ReadOnly]  long          PayloadLength;
```

Description:
Verify나 VerifyMatch Method 호출 후 인증이 성공한 경우 FIR 내부에 담겨있던 Payload를 얻어 올 수 있는데, Binary stream 형태의 Payload 데이터에 대한 길이 값이 담겨있다.

Related methods:
Verify, VerifyMatch

Related properties:
Payload

■ **TextPayload**

Prototype:

```
[ReadOnly]  BSTR          TextPayload;
```

Description:
Verify나 VerifyMatch Method 호출 후 인증이 성공한 경우 FIR 내부에 담겨있던 Payload를 얻어 올 수 있는데, 그 Payload 값이 Text string 형태의 문자열로 담겨있다.

Related methods:
Verify, VerifyMatch

6.5. IFPData Interface

지문 데이터를 얻어오거나 변환하는 기능과 관련된 인터페이스. 지문 데이터를 다른 형태로 변환하고자 할 경우 이 인터페이스를 얻어와 사용해야 한다.

6.5.1. Methods

IFPData Interface의 각종 Method에 대해 설명한다.

■ **ClearFPData**

Prototype:

```
HRESULT ClearFPData();
```

Description:

FPData의 모든 변수를 초기화한다

Parameters:

없음

Related properties:

All

■ **Export**

Prototype:

```
HRESULT Export (VARIANT storedFIR, nDestFPDataType);
```

Description:

FIR 데이터로부터 원하는 형태의 Template 정보를 얻어 온다. Method 호출 후 각각의 Template 정보를 각종 Property를 이용해 얻을 수 있다.
일반적으로 FIR 데이터는 내부 데이터를 암호화 해 숨기고 있기 때문에 각 Template 별로 정보를 알 수는 없다. 때문에 다른 응용 프로그램이나 단말기와 같은 곳에서 Template 별로 데이터를 처리해야 하는 경우에는 이 Method를 이용해 데이터를 변환해야지만 사용할 수 있다.

Parameters:

storedFIR:
변환을 요하는 FIR 데이터. Binary stream형태의 FIR 데이터일수도 있고 Text string형태의 TextFIR 데이터일수도 있다. 어떤 것을 사용하건 상관없다.

nDestFPDataType:
변환 할 Template 데이터의 Type을 지정.

가질 수 있는 값은 UCBioAPI_TEMPLATE_TYPE 정의 참조.

Related properties:
ErrorCode, ErrorDescription
TotalFingerCount, SampleNumber, FingerID, FPSampleData, FPSampleDataLength

■ **Import**

Prototype:

```
HRESULT Import (
            BOOL bInitialize,
            long nFingerID,
            long nPurpose,
            long nSrcFPDataType,
            long nFPDataSize,
            VARIANT FPData1,
            [in, optional] VARIANT FPData2);
```

Description:

특정 형태를 가지는 한 개 또는 다수개의 Template 정보를 이용해 FIR 데이터를 만든다. Method 호출 후 Template 정보가 담긴 FIR을 각종 Property를 통해 얻을 수 있다.
일반적으로 UCBioBSP SDK는 Template을 이용해 바로 인증을 수행 할 수는 없도록 되어 있으며 모든 데이터는 FIR 형태로 변환을 거친 후 사용 할 수 있도록 되어 있다. 그러므로 이 Method를 이용해 FIR 데이터로 변환하는 작업이 필요하다.
이렇게 만들어진 데이터는 FIR이나 TextFIR Property를 이용해 얻을 수 있다.

Parameters:

bInitialize
FIR 데이터를 초기화 하고 새로 만들 것인지를 지정.
만약 이 값이 False이면 지금 추가하는 Template 데이터는 내부적으로 만들어진 FIR 데이터에 계속 추가되게 되어 다수개의 Template 데이터를 가지는 하나의 FIR 데이터가 만들어지게 된다. 하지만 이 값을 True로 하게 되면 기존에 있던 FIR을 모두 지우고 새로 만들게 된다.

nFingerID:
추가하고자 하는 Template의 손가락 ID 정보. 관련 값은 UCBioAPI_FINGER_ID 참조.

nPurpose:
변환 할 FIR 데이터의 Purpose 값을 지정 할 수 있다.
이 값들은 FIR 데이터에 대한 참조용 값으로만 사용되고 인증 시에는 영향을 미치지 않는다. 좀 더 자세한 설명은 UCBioAPI_FIR_PURPOSE 정의 참조.

nSrcFPDataType:
추가하고자 하는 Template의 Type 정보. 관련 값은 UCBioAPI_TEMPLATE_TYPE 참조.

FPData1:
추가하고자 하는 Template 데이터. (Binary stream 데이터)

FPData2:
추가하고자 하는 손가락의 두 번째 Template 데이터. (Binary stream 데이터)
이 값은 옵션으로 지정하지 않아도 된다. 그렇게 될 경우 생성된 FIR은 내부적으로 SamplesPerFinger의 값이 1이 된다.

Related properties:
ErrorCode, ErrorDescription
FIR, FIRLength, TextFIR

■ **CreateTemplate**

Prototype:

```
HRESULT CreateTemplate (
            VARIANT capturedFIR,
            VARIANT storedFIR,
            [in, optional] VARIANT payload);
```

Description:

기존의 FIR 데이터에 새로운 FIR 데이터를 Merge하거나 새로운 FIR 데이터로 교체하여 새로운 FIR Handle을 만든다. 또한 새로운 Payload를 기존 FIR에 교체 할 경우에도 사용 가능하다. 이렇게 만들어진 데이터는 FIR이나 TextFIR Property를 이용해 얻을 수 있다.

Parameters:

capturedFIR:

새롭게 교체 할 FIR 데이터. Binary stream형태의 FIR 데이터일수도 있고 Text string형태의 TextFIR 데이터일수도 있다. 어떤 것을 사용하건 상관없다.

storedFIR:

베이스로 사용 할 FIR 데이터. Binary stream형태의 FIR 데이터일수도 있고 Text string형태의 TextFIR 데이터일수도 있다. 어떤 것을 사용하건 상관없다.
만약 이 값이 Null이 아니라면 이 값을 베이스로 해서 FIR 데이터가 만들어진다.
즉, 만약 이 FIR 값이 오른손 엄지, 검지, 중지의 데이터를 가지고 있고 새롭게 교체할 capturedFIR 데이터가 오른손 엄지와 왼손 엄지 데이터를 가지고 있다면 최종적으로 만들어지는 FIR 데이터는 storedFIR 데이터에 있던 오른손 검지, 중지와 capturedFIR 데이터에 있던 오른손 엄지, 왼손 엄지 데이터를 모두 가지는 데이터로 만들어 진다. 즉, 기존 데이터에서 새로운 데이터가 추가 및 변경되어 새로운 데이터로 만들어 지게 된다.

payload:

만들어질 FIR 데이터에 담겨질 Payload 데이터. Binary stream형태의 Payload 데이터일수도 있고 Text string형태의 TextPayload 데이터일수도 있다. 어떤 것을 사용하건 상관없다.
이 값은 옵션으로 지정하지 않을 수 있고 지정하지 않을 경우 기존 데이터의 Payload 값이 유지된다.

Related properties:

ErrorCode, ErrorDescription
FIR, FIRLength, TextFIR

6.5.2. Properties

IFPData Interface의 각종 Property에 대해 설명한다.

■ **ErrorCode**

Prototype:

```
[ReadOnly]  long          ErrorCode;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.

■ **ErrorDescription**

Prototype:

```
[ReadOnly]  BSTR          ErrorDescription;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 문자열로 담겨있다.
ErrorCode에 대한 오류 값을 문자열로 출력하고자 할 때 사용할 수 있다.

■ **TotalFingerCount**

Prototype:

```
[ReadOnly]  long          TotalFingerCount;
```

Description:

변환된 FIR의 총 손가락 개수를 담고 있다.
반드시 Export Method 호출 후 사용하여야 한다.

Related methods:

Export

Related properties:

FingerID

■ **FingerID**

Prototype:

```
[ReadOnly]  long          FingerID(long nIndex);
```

Description:

변환된 FIR의 손가락 ID 정보를 배열로 담고 있다.
nIndex는 0부터 (TotalFingerCount – 1)의 값을 가질 수 있다.
반드시 Export Method 호출 후 사용하여야 한다.

Related methods:
Export

Related properties:
FPSampleDataLength, FPSampleData

■ **SampleNumber**

Prototype:

```
[ReadOnly]  long          SampleNumber;
```

Description:
변환된 FIR의 손가락별 Template의 개수를 담고 있다. 1또는 2의 값이 담기게 된다.
반드시 Export Method 호출 후 사용하여야 한다.

Related methods:
Export

Related properties:
FPSampleDataLength, FPSampleData

■ **FPSampleData**

Prototype:

```
[ReadOnly]  VARIANT       FPSampleData(
                                long nFingerID,
                                long SampleNum);
```

Description:
변환된 FIR의 손가락별 Template의 Binary stream 데이터를 얻는다.
nFingerID와 SampleNum은 FingerID 및 SampleNumber Property를 이용해 얻을 수 있다.
Binary stream 데이터의 길이 값은 FPSampleDataLength Property를 이용해 얻을 수 있다.
반드시 Export Method 호출 후 사용하여야 한다.

Parameters:
nFingerID:
얻어올 손가락 ID 번호.

nSampleNumber:
얻어올 Sample 번호. 0또는 1의 값을 사용한다.

Related methods:
Export

Related properties:
FingerID, SampleNumber, FPSampleDataLength

■ **FPSampleDataLength**

Prototype:

```
[ReadOnly]  long          FPSampleDataLength(
                                long nFingerID,
                                long SampleNum);
```

Description:
변환된 FIR의 손가락별 Template의 데이터 크기를 얻는다.
nFingerID와 SampleNum은 FingerID 및 SampleNumber Property를 이용해 얻을 수 있다.
반드시 Export Method 호출 후 사용하여야 한다.

Parameters:
nFingerID:
얻어올 손가락 ID 번호.

nSampleNumber:
얻어올 Sample 번호. 0또는 1의 값을 사용한다.

Related methods:
Export

Related properties:
FingerID, SampleNumber, FPSampleData

■ **FIR**

Prototype:

```
[ReadOnly]  VARIANT       FIR;
```

Description:

Import 또는 CreateTemplate 등과 같은 Method 호출 후 획득된 FIR 데이터를 Binary stream 데이터로 얻어온다.
데이터의 길이 값은 FIRLength Property에 담겨있다.
얻어오기 위한 FIR Format은 미리 FIRFormat property에 지정해 두어야 한다.

Related methods:
Import, CreateTemplate

Related properties:
FIRLength, FIRFormat

■ **FIRLength**

Prototype:

```
[ReadOnly]  long          FIRLength;
```

Description:
Import 또는 CreateTemplate 등과 같은 Method 호출 후 획득된 Binary stream FIR 데이터의 길이 값을 얻어온다.

Related methods:
Import, CreateTemplate

Related properties:
FIR

■ **TextFIR**

Prototype:

```
[ReadOnly]  BSTR          TextFIR;
```

Description:
Import 또는 CreateTemplate 등과 같은 Method 호출 후 획득된 FIR 데이터를 Text string 형태의 문자열로 얻어온다. 문자열이므로 길이 값은 따로 지정하지 않는다.
얻어오기 위한 FIR Format은 미리 FIRFormat property에 지정해 두어야 한다.

Related methods:
Import, CreateTemplate

Related properties:
FIRFormat

■ **FIRFormat**

Prototype:

```
[Read/Write] long        FIRFormat;
```

Description:

얻어올 FIR의 데이터 포맷을 지정한다. 향후 FIR 데이터의 구조가 바뀔 경우 이 포맷값이 바뀌어 하위 호환성을 유지 할 수 있다. 현재는 STANDARD(1) 포맷만 지원하고 있다.

Related properties:

FIR, TextFIR

6.6. IFPImage Interface

지문 데이터를 이미지로 얻어오거나 이미지 파일로 저장하는 기능과 관련된 인터페이스. 지문 데이터를 이미지로 변환 하고자 할 경우 이 인터페이스를 얻어와 사용해야 한다.

6.6.1. Methods

IFPImage Interface의 각종 Method에 대해 설명한다.

■ **Export**

Prototype:

```
HRESULT Export ();
```

Description:

가장 최근에 획득된 Audit FIR 데이터로부터 이미지 데이터를 추출한다. Method 호출 후 각각의 이미지 정보를 각종 Property를 이용해 얻을 수 있다.
일반적으로 FIR 데이터는 내부 데이터를 암호화 해 숨기고 있기 때문에 각 손가락 별로 이미지 정보를 알 수는 없다. 때문에 손가락 별로 이미지를 저장해야 하는 경우에는 이 Method를 이용해 데이터를 변환해야지만 사용 할 수 있다.

Related properties:

ErrorCode, ErrorDescription
TotalFingerCount, SampleNumber, FingerID, RawData, ImageWidth, ImageHeight

Related methods:

IDevice.Capture, IExtraction.Enroll, IExtraction.Verify

■ **ExportEx**

Prototype:

```
HRESULT ExportEx (VARIANT capturedAuditFIR);
```

Description:

주어진 Audit FIR 데이터로부터 이미지 데이터를 추출한다. Method 호출 후 각각의 이미지 정보를 각종 Property를 이용해 얻을 수 있다.
일반적으로 FIR 데이터는 내부 데이터를 암호화 해 숨기고 있기 때문에 각 손가락 별로 이미지 정보를 알 수는 없다. 때문에 손가락 별로 이미지를 저장해야 하는 경우에는 이 Method를 이용해 데이터를 변환해야지만 사용 할 수 있다.
Export Method와 동일하지만 Export Method는 가장 최근에 획득된 Audit FIR 데이터를 자동으로 사용하지만 이 Method는 주어진 Audit FIR 데이터를 이용한다는 것이 다르다.

Parameters:

capturedAuditFIR:

변환 할 Audit FIR 데이터. Binary stream형태의 Audit FIR 데이터일수도 있고 Text string형태의 TextAuditData 데이터일수도 있다. 어떤 것을 사용하건 상관없다.

Related properties:
ErrorCode, ErrorDescription
TotalFingerCount, SampleNumber, FingerID, RawData, ImageWidth, ImageHeight, AuditData, AuditDataLength, TextAuditData

■ **Save**

Prototype:

```
HRESULT Save (
            BSTR bszImgFilePath,
            long nImageType,
            long nFingerID,
            [in,optional] long nSampleNumber);
```

Description:
Export된 Audit용 이미지 데이터를 원하는 형태의 이미지 포맷의 파일로 저장한다.
nFingerID와 nSampleNumber는 FingerID 및 SampleNumber Property를 이용해 얻을 수 있다.
이 Method는 반드시 Export나 ExportEx Method 호출 후에 사용하여야 한다.

Parameters:
bszImgFilePath:
저장할 이미지 파일을 Full path로 지정.

nImageType:
저장할 이미지 파일의 포맷을 지정.
사용 가능한 값은 다음과 같다.

```
1 -       RAW
2 -       BMP
3 -       JPG
```

nFingerID:
저장할 손가락 ID 번호.

nSampleNumber:
저장할 Sample 번호. 0또는 1의 값을 사용한다.
이 값은 옵션으로 만약 지정하지 않을 경우 0의 값이 사용된다.

Related properties:
ErrorCode, ErrorDescription
TotalFingerCount, SampleNumber, FingerID

Related methods:
Export, ExportEx

6.6.2. Properties

IFPImage Interface의 각종 Property에 대해 설명한다.

■ **ErrorCode**

Prototype:

```
[ReadOnly]  long          ErrorCode;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.

■ **ErrorDescription**

Prototype:

```
[ReadOnly]  BSTR          ErrorDescription;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 문자열로 담겨있다.
ErrorCode에 대한 오류 값을 문자열로 출력하고자 할 때 사용할 수 있다.

■ **TotalFingerCount**

Prototype:

```
[ReadOnly]  long          TotalFingerCount;
```

Description:

변환된 Audit FIR의 총 손가락 개수를 담고 있다.
반드시 Export 또는 ExportEx Method 호출 후 사용하여야 한다.

Related methods:

Export, ExportEx

Related properties:

FingerID

■ **FingerID**

Prototype:

```
[ReadOnly]  long          FingerID(long nIndex);
```

Description:

변환된 Audit FIR의 손가락 ID 정보를 배열로 담고 있다.
nIndex는 0부터 (TotalFingerCount – 1)의 값을 가질 수 있다.
반드시 Export 또는 ExportEx Method 호출 후 사용하여야 한다.

Related methods:
Export, ExportEx, Save

Related properties:
RawData

■ **SampleNumber**

Prototype:

```
[ReadOnly]  long          SampleNumber;
```

Description:
변환된 Audit FIR의 손가락별 Template의 개수를 담고 있다. 1또는 2의 값이 담기게 된다.
반드시 Export 또는 ExportEx Method 호출 후 사용하여야 한다.

Related methods:
Export, ExportEx, Save

Related properties:
RawData

■ **ImageWidth / ImageHeight**

Prototype:

```
[ReadOnly]  long          ImageWidth;
[ReadOnly]  long          ImageHeight;
```

Description:
변환된 Audit FIR의 이미지 크기를 담고 있다.
반드시 Export 또는 ExportEx Method 호출 후 사용하여야 한다.

Related methods:
Export, ExportEx

Related properties:
RawData

■ **RawData**

Prototype:

```
[ReadOnly]  VARIANT       RawData (
                                long nFingerID,
                                [in, optional] long nSampleNumber);
```

Description:

변환된 Audit FIR의 Raw 이미지 데이터를 Binary stream으로 얻어온다.
nFingerID와 nSampleNumber는 FingerID 및 SampleNumber Property를 이용해 얻을 수 있다.
반드시 Export 또는 ExportEx Method 호출 후 사용하여야 한다.

Related methods:

Export, ExportEx

Related properties:

FingerID, SampleNumber

■ **AuditData**

Prototype:

```
[ReadOnly]  VARIANT       AuditData
```

Description:

가장 최근에 획득된 Audit FIR 데이터를 Binary stream으로 얻어온다.
이렇게 얻어진 데이터는 ExportEx Method의 인자로 사용 할 수 있다.
이 데이터를 얻기 위해서는 이전에 IDevice.Capture, IExtraction.Enroll, IExtraction.Verify 와 같은 Method 들이 수행되어야 한다.
Binary stream의 길이는 AuditDataLength Property에 담겨 있다.

Related methods:

IDevice.Capture, IExtraction.Enroll, IExtraction.Verify, ExportEx

Related properties:

AuditDataLength

■ **AuditDataLength**

Prototype:

```
[ReadOnly]  long          AuditDataLength
```

Description:
가장 최근에 획득된 Audit FIR 데이터의 Binary stream에 대한 길이 값을 얻어온다.
이 데이터를 얻기 위해서는 이전에 IDevice.Capture, IExtraction.Enroll, IExtraction.Verify 와 같은 Method 들이 수행되어야 한다.

Related methods:
IDevice.Capture, IExtraction.Enroll, IExtraction.Verify, ExportEx

Related properties:
AuditData

■ **TextAuditData**

Prototype:

```
[ReadOnly]  BSTR          TextAuditData
```

Description:
가장 최근에 획득된 Audit FIR 데이터를 Text string 형태의 문자열로 얻어온다.
이렇게 얻어진 데이터는 ExportEx Method의 인자로 사용 할 수 있다.
이 데이터를 얻기 위해서는 이전에 IDevice.Capture, IExtraction.Enroll, IExtraction.Verify 와 같은 Method 들이 수행되어야 한다.

Related methods:
IDevice.Capture, IExtraction.Enroll, IExtraction.Verify, ExportEx

6.7. IFastSearch Interface

1:N 고속 인증을 수행하기 위한 FastSearch 기능과 관련된 인터페이스. FastSearch 관련 DB 관리 및 인증 수행 등과 같은 기능을 사용하고자 할 경우 이 인터페이스를 얻어와 사용해야 한다.

6.7.1. Methods

IFastSearch Interface의 각종 Method에 대해 설명한다.

■ **AddFIR**

Prototype:

```
HRESULT AddFIR (VARIANT FIR, long nUserID);
```

Description:

FastSearch용 DB에 FIR 데이터를 추가 한다.
1:N 인증을 위해서는 우선 1:N을 수행 할 DB를 구축해야 하는데 이 함수를 이용해 그 DB를 만들게 된다.
또한 1:N 인증은 내부적으로 FIR 단위가 아니라 Template 단위로 동작하게 되어있다. 따라서 1개의 FIR을 추가하더라도 FIR 내부에 여러 개의 Template이 존재할 경우 DB에는 여러 개의 데이터가 추가되게 된다. 추가되는 Template에 대한 정보는 AddedFpInfo 등의 Property를 통해 얻을 수 있다.

Parameters:

FIR
추가할 FIR 데이터. Binary stream형태의 FIR 데이터일수도 있고 Text string형태의 TextFIR 데이터일수도 있다. 어떤 것을 사용하건 상관없다.

nUserID:
추가할 FIR의 사용자 ID 값.

Related properties:

ErrorCode, ErrorDescription
FpCount, IsFpExisted, AddedFpCount, AddedFpInfo, FpInfo

Related methods:

RemoveFp, RemoveUser, ClearDB, SaveDBToFile, LoadDBFromFile

■ **RemoveFp**

Prototype:

```
HRESULT RemoveFp (long nUserID, long nFingerID, long nSampleNumber);
```

Description:
FastSearch용 DB에서 특정 데이터를 삭제한다.

Parameters:
nUserID:
삭제할 Template의 사용자 ID 정보

nFingerID:
삭제할 Template의 손가락 ID 정보

nSampleNumber:
삭제할 Template의 Sample 번호

Related properties:
ErrorCode, ErrorDescription
FpCount, IsFpExisted, FpInfo

Related methods:
AddFIR, RemoveUser, ClearDB, SaveDBToFile, LoadDBFromFile

■ **RemoveUser**

Prototype:

```
HRESULT RemoveUser (long nUserID);
```

Description:
FastSearch용 DB에서 특정 사용자 ID용 데이터를 모두 삭제한다.
UCBioAPI_RemoveFpFromFastSearchDB 함수와 비슷하지만 동일 사용자 ID로 여러 개의 Template 데이터가 DB에 들어 있는 경우 모두 삭제하는 것이 다르다.

Parameters:
nUserID:
삭제할 Template의 사용자 ID 정보

Related properties:
ErrorCode, ErrorDescription
FpCount, IsFpExisted, FpInfo

Related methods:
AddFIR, RemoveFp, ClearDB, SaveDBToFile, LoadDBFromFile

■ **ClearDB**

Prototype:

```
HRESULT ClearDB ();
```

Description:

FastSearch용 DB를 메모리 해제 한다.

Related properties:

ErrorCode, ErrorDescription
FpCount, IsFpExisted, FpInfo

Related methods:

AddFIR, RemoveFp, RemoveUser, SaveDBToFile, LoadDBFromFile

■ **SaveDBToFile**

Prototype:

```
HRESULT SaveDBToFile(BSTR bszFilePath);
```

Description:

FastSearch용 DB를 파일로 저장 한다. 이렇게 메모리 DB를 파일로 저장하게 되면 다음 번에 사용 시 LoadDBFromFile Method를 이용해 빠르게 로드해 사용 할 수 있게 된다.

Parameters:

bszFilePath:
파일로 저장할 파일명의 Full path 값.

Related properties:

ErrorCode, ErrorDescription
FpCount, IsFpExisted, FpInfo

Related methods:

AddFIR, RemoveFp, RemoveUser, ClearDB, LoadDBFromFile

■ **LoadDBFromFile**

Prototype:

```
HRESULT LoadDBFromFile(BSTR bszFilePath);
```

Description:

FastSearch용 DB를 파일로부터 메모리로 읽어온다.
이렇게 읽어올 수 있는 파일은 SaveDBToFile Method를 이용해 저장된 파일만 가능하다.

Parameters:
bszFilePath:
DB로 불러올 파일명의 Full path 값.

Related properties:
ErrorCode, ErrorDescription
FpCount, IsFpExisted, FpInfo

Related methods:
AddFIR, RemoveFp, RemoveUser, ClearDB, SaveDBToFile

■ IdentifyUser

Prototype:

```
HRESULT IdentifyUser(VARIANT processedFIR, long nSecuLevel);
```

Description:
FastSearch용 DB에서 특정 FIR과의 1:N 인증을 시도한다.
함수 호출 후 인증이 성공하면 인증된 Template의 정보를 얻을 수 있다. 1:N의 특성상 인증 시간은 매번 달라 질 수 있으며 시스템 속도와 메모리에 따라서도 달라 질 수 있다.
인증 시간이 길어질 수 있으므로 최대 인증 시간을 지정하고자 할 경우 MaxSearchTime Property를 지정하면 된다.

Parameters:
processedFIR:
인증할 FIR 데이터. Binary stream형태의 FIR 데이터일수도 있고 Text string형태의 TextFIR 데이터일수도 있다. 어떤 것을 사용하건 상관없다.

nSecuLevel:
인증 시 사용할 보안 수준을 지정.
가질 수 있는 값은 UCBioAPI_FIR_SECURITY_LEVEL 정의 참조.

Related properties:
ErrorCode, ErrorDescription
MatchedFpInfo, MaxSearchTime, UseGroupMatch, MatchMethod

6.7.2. Properties

IFastSearch Interface의 각종 Property에 대해 설명한다.

■ **ErrorCode**

Prototype:

```
[ReadOnly]  long          ErrorCode;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.

■ **ErrorDescription**

Prototype:

```
[ReadOnly]  BSTR          ErrorDescription;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 문자열로 담겨있다.
ErrorCode에 대한 오류 값을 문자열로 출력하고자 할 때 사용할 수 있다.

■ **FpCount**

Prototype:

```
[ReadOnly]  long          FpCount;
```

Description:

현재 FastSearch DB에 담겨 있는 Template의 개수를 담고 있다.

Related methods:

AddFIR, RemoveFp, RemoveUser, ClearDB, SaveDBToFile, LoadDBFromFile

■ **FpInfo**

Prototype:

```
[ReadOnly]  VARIANT       FpInfo(long index);
```

Description:

현재 FastSearch DB에 들어있는 Template의 정보를 얻어온다.
얻어온 정보는 ITemplateInfo 인터페이스로 얻어진다.
Index 값은 0부터 (FpCount – 1)의 값을 가질 수 있다.

Related properties:
ITemplateInfo.UserID, ITemplateInfo.FingerID, ITemplateInfo.SampleNumber

■ **IsFpExisted**

Prototype:

```
[ReadOnly]  BOOL          IsFpExisted(
                                long nUserID,
                                long nFingerID,
                                long nSampleNumber);
```

Description:
현재 FastSearch DB에 지정한 정보를 가지는 Template이 있는지를 검사.

Parameters:
nUserID:
삭제할 Template의 사용자 ID 정보

nFingerID:
삭제할 Template의 손가락 ID 정보

nSampleNumber:
삭제할 Template의 Sample 번호

Related methods:
AddFIR, RemoveFp, RemoveUser, ClearDB, SaveDBToFile, LoadDBFromFile

■ **AddedFpCount**

Prototype:

```
[ReadOnly]  BOOL          AddedFpCount;
```

Description:
AddFIR Method를 통해 FIR 데이터를 FastSearch DB에 추가 할 경우 추가된 Template의 개수를 얻어온다.
1:N 인증은 내부적으로 FIR 단위가 아니라 Template 단위로 동작하게 되어있다. 따라서 1개의 FIR을 추가하더라도 FIR 내부에 여러 개의 Template이 존재할 경우 DB에는 여러 개의 데이터가 추가되게 된다. 이 Property에는 그렇게 추가된 개수가 담겨있게 된다.
반드시 AddFIR 호출 후에 사용하여야 한다.

Related methods:

AddFIR

■ **AddedFpInfo**

Prototype:

```
[ReadOnly]  VARIANT      AddedFpInfo(long index);
```

Description:

AddFIR Method를 통해 FIR 데이터를 FastSearch DB에 추가 할 경우 추가된 Template의 각각의 정보를 얻어온다. 얻어온 정보는 ITemplateInfo 인터페이스로 얻어진다.
Index 값은 0부터 (AddedFpCount – 1)의 값을 가질 수 있다.
반드시 AddFIR 호출 후에 사용하여야 한다.

Related methods:

AddFIR

Related properties:

ITemplateInfo.UserID, ITemplateInfo.FingerID, ITemplateInfo.SampleNumber

■ **MatchedFpInfo**

Prototype:

```
[ReadOnly]  VARIANT      MatchedFpInfo;
```

Description:

IdentifyUser Method를 통해 1:N 인증을 성공한 경우 인증된 Template의 정보를 얻어온다. 얻어온 정보는 ITemplateInfo 인터페이스로 얻어진다.
반드시 IdentifyUser 호출 후에 사용하여야 한다.

Related methods:

IdentifyUser

Related properties:

ITemplateInfo.UserID, ITemplateInfo.FingerID, ITemplateInfo.SampleNumber

■ **MaxSearchTime**

Prototype:

```
[Read/Write] long        MaxSearchTime;
```

Description:

IdentifyUser Method를 통해 1:N 인증을 할 경우 DB 크기와 시스템 성능에 따라 인증 속도는 달라지게 된다. 만약 인증 속도에 의해 인증 시간이 너무 오래 걸릴 것을 막기 위해 최대 인증 시간을 지정해 그 시간 안에서만 인증을 수행 할 수 있다. 만약 인증 시간 안에 인증이 되지 못한 경우에는 IdentifyUser가 오류를 리턴한다.
이 값의 단위는 millisecond이므로 10초를 지정할 경우 10,000이라는 값을 주어야 한다. 만약 이 값이 0일 경우 최대 인증 시간을 지정하지 않는 것과 같다. 기본 값은 0이다.

Related methods:
IdentifyUser

■ **UseGroupMatch**

Prototype:

```
[Read/Write] long        UseGroupMatch;
```

Description:
IdentifyUser Method를 통해 1:N 인증을 할 경우 그룹단위의 인증을 할 것인지를 결정한다. 0일 경우 DB에 들어있는 순서대로 인증을 수행하고 1일 경우 그룹단위의 인증을 수행한다. 기본값은 1이다.
이 값은 가능한 1로 설정하고 인증을 하는 것이 좋다.

Related methods:
IdentifyUser

■ **MatchMethod**

Prototype:

```
[Read/Write] long        MatchMethod;
```

Description:
IdentifyUser Method를 통해 1:N 인증을 할 경우 인증을 수행할 방법을 결정한다. 0일 경우 설정된 인증 레벨을 이용해 그 레벨을 넘는 것이 나오면 바로 인증을 종료하는 방법이고 1일 경우 최고점 인증 방법으로 가장 인증 레벨이 높은 값을 찾는 방식이다. 기본값은 0이다.
이 값은 가능한 0으로 설정하고 인증을 하는 것이 좋다.

Related methods:
IdentifyUser

6.8. ITemplateInfo Interface

이 Interface는 IFastSearch Interface에서 Template 정보를 표현하기 위해 사용된다. Method는 없이 Property로만 이루어져 있다.

6.8.1. Properties

ITemplateInfo Interface의 각종 Property에 대해 설명한다.

■ **UserID**

Prototype:

```
[ReadOnly]  long          UserID;
```

Description:

FastSearch의 Template 정보 중 사용자 ID 값을 가진다.

■ **FingerID**

Prototype:

```
[ReadOnly]  long          FingerID;
```

Description:

FastSearch의 Template 정보 중 손가락 ID 값을 가진다. UCBioAPI_FINGER_ID 참조.

■ **SampleNumber**

Prototype:

```
[ReadOnly]  long          SampleNumber;
```

Description:

FastSearch의 Template 정보 중 Sample 번호를 가진다. 0또는 1의 값을 가진다.

6.9. ISmartCard Interface

스마트카드와 관련된 인터페이스. 스마트카드에 데이터를 저장 및 읽어오는 기능 등을 사용하기 위해 이 인터페이스를 얻어와 사용해야 한다.

- **참고 - 스마트카드 사용을 위한 함수들은 디바이스의 Firmware 버전에 따라 지원하지 않는 Method가 있을 수 있다.**

6.9.1. Methods

ISmartCard Interface의 각종 Method에 대해 설명한다.

■ **RFPowerOn**

Prototype:

```
HRESULT RFPowerOn ();
```

Description:

5초 이내에 RF 영역의 RF Power를 ON 한다.
대부분의 스마트카드 API 들은 RF Power가 On이 된 상태에서 동작한다.

Related properties:

ErrorCode, ErrorDescription
LED

Related methods:

IDevice.Open, RFPowerOff

■ **RFPowerOff**

Prototype:

```
HRESULT RFPowerOff ();
```

Description:

5초 이내에 RF 영역의 RF Power를 OFF 한다.

Related properties:

ErrorCode, ErrorDescription
LED

Related methods:

RFPowerOn

■ **RFFunction**

Prototype:

```
HRESULT RFFunction(VARIANT CmdBuff, long nCmdLen);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 명령어를 전송하고 그 결과를 받아 온다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.

Parameters:

CmdBuff:
전송할 명령어가 담긴 버퍼 포인터.

nCmdLen:
전송할 명령어가 담긴 버퍼 포인터의 길이.

Related properties:

ErrorCode, ErrorDescription
LED

Related methods:

RFPowerOn

■ **ReadSerial**

Prototype:

```
HRESULT ReadSerial ();
```

Description:

5초 이내에 RF 영역에 카드가 있으면 해당 시리얼넘버를 얻어온다.
이 명령어는 내부에 RF Power를 ON 하는 기능이 있으므로 함수 호출 전에 RFPowerOn Method를 호출 할 필요가 없다.
함수 호출 후 ResultBuffer나 Serial Property를 통해 Serial 값을 얻을 수 있다.
(ResultBuffer를 스트링으로 받고자 하는 경우는 textSerial Property 사용하면 됨)

Related properties:

ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength, Serial, textSerial

■ **ReadBlock**

Prototype:

```
HRESULT ReadBlock (long SectorNum, long BlockNum);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록의 데이터를 읽어온다.
반드시 RF Power가 ON이 되어 있어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
읽고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
읽고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB, ResultBuffer, ResultLength

Related methods:

RFPowerOn, WriteBlock

■ **WriteBlock**

Prototype:

```
HRESULT WriteBlock (long SectorNum, long BlockNum, VARIANT varData);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록에 16bytes 길이의 데이터를 기록한다.
반드시 RF Power가 ON이 되어 있어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
쓰고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

varData:
쓰고자 하는 16bytes의 블록 데이터.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB

Related methods:

RFPowerOn, ReadBlock

■ **ReadSector**

Prototype:

```
HRESULT ReadSector (long SectorNum);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector의 데이터를 읽어온다.
반드시 RF Power가 ON이 되어 있어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
읽고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB, ResultBuffer, ResultLength

Related methods:

RFPowerOn, WriteSector

■ **WriteSector**

Prototype:

```
HRESULT WriteSector (long SectorNum, VARIANT varData48);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector에 48bytes 길이의 데이터를 기록한다.
반드시 RF Power가 ON이 되어 있어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

varData:
쓰고자 하는 48bytes의 Sector 데이터.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB

Related methods:

RFPowerOn, ReadSector

■ **ReadSectorFieldContent**

Prototype:

```
HRESULT ReadSectorFieldContent (
                    long StartSectorNum,
                    long EndSectorNum);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector 구간 안의 모든 데이터를 읽어온다.
지정 가능한 구간은 최대 10개 Sector이며 최대 얻어 올 수 있는 데이터 길이는 480bytes 까지다. (48 Bytes * 10 Sectors = 480 Bytes)
반드시 RF Power가 ON이 되어 있어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

StartSectorNum:
읽고자 하는 Sector의 시작 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

EndSectorNum:
읽고자 하는 Sector의 끝 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB, ResultBuffer, ResultLength

Related methods:

RFPowerOn, WriteSectorFieldContent

■ **WriteSectorFieldContent**

Prototype:

```
HRESULT WriteSectorFieldContent (
                        long StartSectorNum,
                        long EndSectorNum,
                        VARIANT varData);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector 구간 안의 모든 데이터에 기록한다.
지정 가능한 구간은 최대 10개 Sector이며 최대 기록 할 수 있는 데이터 길이는 480bytes 까지다. (48 Bytes * 10 Sectors = 480 Bytes)
반드시 RF Power가 ON이 되어 있어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

StartSectorNum:
쓰고자 하는 Sector의 시작 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

EndSectorNum:
쓰고자 하는 Sector의 끝 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

varData:
쓰고자 하는 데이터. 데이터의 길이는 (EndSectorNum – StartSectorNum + 1) * 48bytes가 된다.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB

Related methods:

RFPowerOn, ReadSectorFieldContent

■ **PreValue**

Prototype:

```
HRESULT PreValue (long SectorNum, long BlockNum, long newVal);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록을 Value 모드로 전환하고 4bytes의 newVal 값을 기록한다.
만약 기록하는 pData의 값이 0x00000000 이라면 함수 호출 후 해당 블록은 다음과 같이 값이 기록되게 된다. (0x00000000FFFFFFFF00000000FFFFFFFF00FF00FF)
이렇게 Value 모드로 변환된 블록은 ReadValue, IncrementValue와 같은 Value 모드용 Method들을 사용 할 수 있게 된다.
Value 모드에 대한 자세한 사항은 이 문서의 SmartCard 사용하기를 참조하면 된다.
반드시 RF Power가 ON이 되어 있어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
쓰고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

newVal:
기록 할 4bytes의 데이터

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB, Value

Related methods:

RFPowerOn, ReadValue, IncrementValue, DecrementValue

■ **ReadValue**

Prototype:

```
HRESULT ReadValue(long SectorNum, long BlockNum);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록의 4bytes Value 데이터를 읽어온다.
반드시 RF Power가 ON이 되어 있어야 하고 지정한 블록이 Value 모드이어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
함수 호출 후 ResultBuffer나 Value Property를 통해 값을 얻을 수 있다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
읽고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
읽고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB, ResultBuffer, ResultLength, Value

Related methods:

RFPowerOn, PreValue, IncrementValue, DecrementValue

■ **IncrementValue**

Prototype:

```
HRESULT IncrementValue(long SectorNum, long BlockNum, long newVal);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록의 4bytes Value 데이터를 지정한 값만큼 증가시킨다.
반드시 RF Power가 ON이 되어 있어야 하고 지정한 블록이 Value 모드이어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
함수 호출 후 ResultBuffer나 Value Property를 통해 값을 얻을 수 있다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
쓰고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB, ResultBuffer, ResultLength, Value

Related methods:

RFPowerOn, PreValue, ReadValue, DecrementValue

■ **DecrementValue**

Prototype:

```
HRESULT DecrementValue(long SectorNum, long BlockNum, long newVal);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 블록의 4bytes Value 데이터를 지정한 값만큼 감소시킨다.
반드시 RF Power가 ON이 되어 있어야 하고 지정한 블록이 Value 모드이어야 한다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
함수 호출 후 ResultBuffer나 Value Property를 통해 값을 얻을 수 있다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

BlockNum:
쓰고자 하는 Block의 번호. 0 ~ 3 사이의 값을 가진다.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB, ResultBuffer, ResultLength, Value

Related methods:

RFPowerOn, PreValue, ReadValue, IncrementValue

■ **WriteSectorTrailer**

Prototype:

```
HRESULT WriteSectorTrailer (
                    long SectorNum,
                    VARIANT NewAccessBit,
                    VARIANT NewKeyA,
                    VARIANT NewKeyB);
```

Description:

5초 이내에 RF 영역에 카드가 있으면 지정한 Sector의 Sector Trailer 영역을 수정한다.
Sector Trailer에는 해당 Sector의 Key A, Access Bits, Key B 값이 들어 있다.
해당 Sector의 Access 권한에 따라 성공, 실패 여부가 결정된다.
접근 권한에 맞는 AuthMode와 Key값이 미리 설정 되어 있어야 한다.
이 함수는 Mifare 카드 관련 Method이다.

Parameters:

SectorNum:
쓰고자 하는 Sector의 번호. 카드의 메모리 크기에 따라 최대 값이 달라질 수 있다.

NewAccessBit:
새로운 Access Bits를 담고 있는 4bytes 길이의 데이터.

NewKeyA:
새로운 Key A를 담고 있는 6bytes 길이의 데이터.

NewKeyB:
새로운 Key B를 담고 있는 6bytes 길이의 데이터.

Related properties:

ErrorCode, ErrorDescription
LED, AuthMode, KeyA, KeyB, ResultBuffer, ResultLength, Value

Related methods:

RFPowerOn, PreValue, ReadValue, IncrementValue

■ **ReqA**

Prototype:

```
HRESULT ReqA();
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 ATQ를 얻는다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Related properties:

ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:

RFPowerOn, HaltA

■ **WupA**

Prototype:

```
HRESULT WupA();
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 ATQ를 얻는다.
ReqA Method와 동일하지만 Halt 상태인 카드도 응답하는 점이 다르다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Related properties:

ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:

RFPowerOn, HaltA

■ **Select**

Prototype:

```
HRESULT Select();
```

Description:
Type A 카드로부터 ATQ를 얻은 상태에서, 5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 카드의 SAK 값과 UID 값을 얻어온다.
SAK는 1byte의 길이 값을 가지고 UID는 가변 길이의 길이 값을 가진다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Related properties:
ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:
RFPowerOn

■ **HaltA**

Prototype:

```
UCBioAPI_RETURN UCBioAPI UCBioAPI_SC_Halt (
            [IN]  UCBioAPI_HANDLE    hHandle,
            [IN]  UCBioAPI_UINT16    wLed);
```

Description:
Select 상태인 Type A 카드를 Halt 상태로 바꾼다.
이렇게 Halt 상태가 된 카드는 ReqA Method에 대해서는 응답을 하지 않고 WupA Method에 대해서만 응답하게 된다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Related properties:
ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:
RFPowerOn, ReqA, WupA

■ **Rats**

Prototype:

```
HRESULT Rats(long fsdi, long cid);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 카드의 ATS(Answer To Select) 값을 얻어온다. ATS를 얻기 위해서는 SAK 값이 0x2X(ISO14443-4 지원)일 때 가능하다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

Fsdi:

FSDI(Frame Size for proximity coupling Device Integer)의 설정을 통해 PCD가 수신 할 수 있는 프레임의 최대 크기를 정할 수 있다. (0x00 ~ 0x0F까지의 값)

cid:

CID(Card Identifier)를 통해 각 개별 카드를 선택적으로 호출하는 것이 가능하다. (0x00 ~ 0x0F까지의 값)

Related properties:

ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:

RFPowerOn

■ **PpsRequest**

Prototype:

```
HRESULT PpsRequest(long cid, long pps0, long pps1);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 카드의 PPSS 값을 얻어온다.
ATS의 값을 통해 변화될 수 있는 Parameter를 지원한다면, PCD에 의해 사용될 수 있다.
즉, ATS 내의 선택적 Parameter인 DS와 DR에서 더 높은 보레이트를 지원하면 양방향으로의 보레이트는 서로 독립적으로 2,4,8배로 증가할 수 있다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

cid:
Rats 호출 시 선택한 CID(Card Identifier) 값. (0x00 ~ 0x0F까지의 값)

pps0:
PPS1을 전송하는지 하지 않는지를 나타낸다.
0x11이면 전송, 0x01이면 전송하지 않음을 의미.

pps1:
상위 2bytes는 DRI이며 하위 2bytes는 DS를 의미.

Related properties:

ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:

RFPowerOn

■ **BlockFormat**

Prototype:

```
HRESULT BlockFormat(
                    long pcb,
                    long CidOrNad,
                    VARIANT inf,
                    long infLen);
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드로부터 데이터를 읽어온다.
데이터 계층에서의 블록 교환에 관한 명령(T=1).
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

pcb:
PCB(Protocol Control Byte)를 의미. (필수)
데이터 전송을 제어하는 데 필요한 정보를 가지고 있음. I – Block, R – Block, S – Block으로 나뉘어짐. 데이터를 읽기 위해서 보통 I – Block을 사용하며 PCB 값은 0x0A, 0x0B, 0x0A 식으로 Bit0을 Toggle하여 보내야 함.

CidOrNad:
Rats로 PICC에 지정한 CID 또는 NAD 값. (선택)

inf:
INF(Information Field)를 의미.
해당 파일을 읽으려면 특별한 포맷을 알고 있어야 함.

infLen:
inf의 길이 값.

Related properties:

ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:

RFPowerOn, Deselect

■ **Deselect**

Prototype:

```
HRESULT Deselect(long CidOrNad);
```

Description:

5초 이내에 RF 영역에 Active 상태인 Type A카드가 있으면 카드를 비활성화 시킨다.
Active State인 카드를 Deselect하면, 카드는 RF영역을 벗어나지 않는 한 BlockFormat에 대한 응답을 하지 않는다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Parameters:

CidOrNad:
Rats로 PICC에 지정한 CID 또는 NAD 값.

Related properties:

ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:

RFPowerOn, BlockFormat

■ **TypeA_ActiveState**

Prototype:

```
HRESULT TypeA_ActiveState();
```

Description:

5초 이내에 RF 영역에 Type A카드가 있으면 카드를 ReqA, Select, Rats까지 한꺼번에 수행해 카드로부터 ATS를 얻어온다.
함수 호출 후 ResultBuffer를 통해 값을 얻을 수 있다.
반드시 RF Power가 ON이 되어 있어야 한다.
이 함수는 ISO14443-A 관련 함수이다.

Related properties:

ErrorCode, ErrorDescription
LED, ResultBuffer, ResultLength

Related methods:

RFPowerOn, ReqA, Select, Rats

6.9.2. Properties

ISmartCard Interface의 각종 Property에 대해 설명한다.

■ **ErrorCode**

Prototype:

```
[ReadOnly]  long          ErrorCode;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 담겨있다.
성공일 경우 0의 값을 가지고 그 이외의 값은 실패를 나타낸다.

■ **ErrorDescription**

Prototype:

```
[ReadOnly]  BSTR          ErrorDescription;
```

Description:

실행한 Method 및 Property 설정 중에 발생한 오류에 대한 값이 문자열로 담겨있다.
ErrorCode에 대한 오류 값을 문자열로 출력하고자 할 때 사용할 수 있다.

■ **LED**

Prototype:

```
[Read/Write] long         LED;
```

Description:

성공이나 실패에 따른 결과를 디바이스의 LED에 표시 할 것인지를 지정.
1을 지정 할 경우 성공 시 디바이스의 LED가 파란색으로 바뀌고 실패 시 LED가 붉은색으로 바뀌게 된다. 0으로 지정 할 경우에는 LED 변화 없음.
기본값은 1이다.

Related methods:

RFFunction, ReadSerial, ReadBlock, WriteBlock, ReadSector, WriteSector, ReadSectorFieldContent, WriteSectorFieldContent, PreValue, ReadValue, IncrementValue, DecrementValue, WriteSectorTrailer, ReqA, WupA, HaltA, Select, Rats, PpsRequest, BlockFormat, Deselect, TypeA_ActiveState

■ **AuthMode**

Prototype:

```
[Read/Write] long         AuthMode;
```

Description:
Mifare 카드 관련 Method에서 Key A 또는 Key B 중 어떤 것을 이용해 인증 할 것인지를 지정. Key A를 사용 할 경우에는 이 값을 0x60으로 지정하고 Key B를 사용 할 경우에는 이 값을 0x61로 지정해야 한다. 기본 값은 0x60이다.

Related methods:
ReadSerial, ReadBlock, WriteBlock, ReadSector, WriteSector, ReadSectorFieldContent, WriteSectorFieldContent, PreValue, ReadValue, IncrementValue, DecrementValue, WriteSectorTrailer

Related properties:
KeyA, KeyB

■ **KeyA / KeyB**

Prototype:

```
[Read/Write] VARIANT     KeyA;
[Read/Write] VARIANT     KeyB;
```

Description:
Mifare 카드 관련 Method에서 사용 할 Key A 또는 Key B의 값을 지정한다.
AuthMode에 따라 KeyA가 사용 될 수도 있고 KeyB가 사용 될 수도 있다.
Key값은 6bytes의 값으로 된 binary 배열이다. 기본 값으로는 "FF FF FF FF FF FF"의 값이 들어가 있다.

Related methods:
ReadSerial, ReadBlock, WriteBlock, ReadSector, WriteSector, ReadSectorFieldContent, WriteSectorFieldContent, PreValue, ReadValue, IncrementValue, DecrementValue, WriteSectorTrailer

Related properties:
AuthMode

■ **ResultBuffer**

Prototype:

```
[ReadOnly] VARIANT       ResultBuffer;
```

Description:
스마트카드 Method들 중 값을 읽기 위한 Method를 호출 후 결과 값이 담겨있는 버퍼 배

열이다. 반드시 Method 호출 후 사용 할 수 있다. 버퍼의 길이 값은 ResultLength Property에 담겨있다.

Related methods:
ReadSerial, ReadBlock, ReadSector, ReadSectorFieldContent, PreValue, ReadValue, IncrementValue, DecrementValue

Related properties:
ResultLength

■ **ResultLength**

Prototype:

```
[ReadOnly] long          ResultLength;
```

Description:
스마트카드 Method들 중 값을 읽기 위한 Method를 호출 후 결과 값이 담겨있는 버퍼 배열의 길이 값이 담겨있다. 반드시 Method 호출 후 사용 할 수 있다.

Related methods:
ReadSerial, ReadBlock, ReadSector, ReadSectorFieldContent, PreValue, ReadValue, IncrementValue, DecrementValue

Related properties:
ResultBuffer

■ **Value**

Prototype:

```
[ReadOnly] long          Value;
```

Description:
Mifare 카드 관련 Method 중에 Value Mode용 Method를 이용해 Value값을 읽을 경우 ResultBuffer를 통해서도 값을 얻을 수 있지만 그와 동일한 값을 이 Value Property를 통해서도 얻을 수 있다.

Related methods:
PreValue, ReadValue, IncrementValue, DecrementValue

Related properties:
ResultBuffer

■ **Serial**

Prototype:

```
[ReadOnly] long          Serial;
```

Description:

Mifare 카드 관련 Method 중에 ReadSerial Method를 이용해 카드의 Serial값을 읽을 경우 ResultBuffer를 통해서도 값을 얻을 수 있지만 그와 동일한 값을 이 Serial Property를 통해서도 얻을 수 있다.

Related methods:

ReadSerial

Related properties:

ResultBuffer

7.Distribution Guide

UCBioBSP SDK를 이용해 응용제품을 개발한 경우 그 제품을 배포 시 같이 배포해야 할 SDK 모듈들에 대해 설명한다.

7.1. 공통 사항

UCBioBSP SDK를 이용해 개발한 제품은 기본적으로 다음과 같은 모듈이 필요하다.

■ **디바이스 드라이버**

설명:

SDK를 사용하기 위해서는 각 디바이스 별로 각각에 맞는 디바이스 드라이버가 설치되어 있어야 한다. 단, 서버에서 인증만 수행 할 경우에는 디바이스가 필요치 않으므로 설치할 필요 없음.

■ **UCDevice.dll**

설치 폴더:

Windows System32 폴더 (Win9x의 경우에는 System 폴더)

설명:

UCBioBSP SDK의 디바이스 제어 모듈로서 디바이스를 사용하는 경우에는 반드시 설치되어야 한다.

■ **UCBioBSP.dll**

설치 폴더:

Windows System32 폴더 (Win9x의 경우에는 System 폴더)

설명:

UCBioBSP SDK의 Core 모듈로서 반드시 설치되어야 한다.

■ **Skin 파일**

설치 폴더:

Windows System32 폴더 (Win9x의 경우에는 System 폴더)

설명:

각 언어별 필요한 Skin 파일을 필요에 따라 설치해 주어야 한다.

7.2. DLL을 이용해 개발한 경우

UCBioBSP.dll만을 이용해 개발한 경우에는 공통 사항에 대해서만 설치하면 된다.

7.3. COM을 이용해 개발한 경우

COM 모듈인 UCBioBSPCOM.dll을 이용해 개발한 경우에는 공통사항에 추가로 다음 파일을 같이 배포하면 된다.

■ **UCBioBSPCOM.dll**

설치 폴더: Bin32 / Bin64

모듈 등록: 반드시 Windows registry에 COM 모듈이 등록되어 있어야 사용 가능.
등록을 위해서는 "regsvr32 UCBioBSPCOM.dll" 이라는 명령을 통해 등록 가능.
또는 인스톨 프로그램에서 Self registration 옵션을 사용해 자동 등록 가능.

7.4. .NET을 이용해 개발한 경우

.NET용 Class library인 UNIONCOMM.SDK.UCBioBSP.dll을 이용해 개발한 경우에는 공통사항에 추가로 다음 파일을 같이 배포하면 된다.

■ **.NET Framework v2.0 이상**

반드시 .NET Framework v2.0 이상이 설치되어 있어야 한다.

■ **UNIONCOMM.SDK.UCBioBSP.dll**

설치 폴더: GAC(Global Assembly Cache) 폴더

모듈 등록: SDK에 같이 포함된 .NET용 설치 파일을 이용해 설치 가능.

7.5. BioAPI Framework 상에서 개발한 경우

BioAPI Framework 상에서 개발한 경우에는 공통사항에 추가로 다음 파일을 같이 배포하면 된다.

■ **BioAPI Framework v2.0 이상**

반드시 BioAPI Framework v2.0 이상이 설치되어 있어야 한다.

■ **UCBioBSP.dll 배포 및 등록**

모듈 등록: BioAPI Framework에서 제공하는 BSP 등록 프로그램을 이용해 반드시 UCBioBSP.dll을 Framework에 등록한 후 사용하여야 한다.

Appendix A. How to enroll fingerprint properly

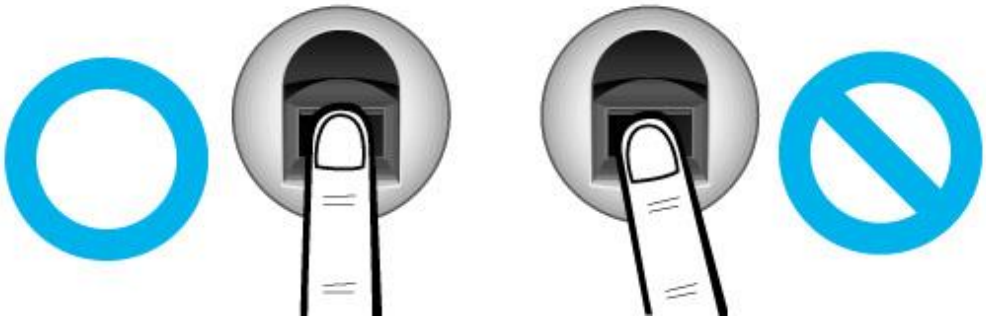
지문 인식기에 올바른 지문을 등록하는 법에 대해 설명한다.
한번 등록된 지문은 계속적으로 인증에 사용되는 경우가 많으므로 처음 한번 잘못 등록한 지문 데이터는 이후 인증율에 계속적으로 나쁜 영향을 미칠 수가 있다. 때문이 좋은 지문을 등록 할 수 있도록 아래 문서를 참조하는 것이 좋다.
만약 SDK를 이용해 개발한 제품을 배포 시 아래 문서도 같이 배포하여 사용자가 지문 입력에 가이드를 제공하는 방법도 좋은 방법이다.

A.1. 올바른 지문 입력 방법

1) 지문의 정 중앙이 지문입력창 중앙에 위치하도록 손가락을 올립니다.



- 입력 시 손가락을 너무 세게 누르지 마시고 손가락이 입력창에 밀착될 정도의 압력만 주십시오.
- 빨간 불이 켜져있는 동안은 손가락을 움직이지 마시고 빨간 불이 꺼진 후 손가락을 떼십시오.
- 지문 입력창에 손가락을 비스듬하지 않게 바르게 올려 놓으십시오.



2) 가급적 검지 손가락의 지문을 입력하십시오.

- 가급적 검지 손가락을 이용하여 손도장을 찍는 기분으로 입력하십시오.
- 검지 손가락을 이용하면 정확하고 안정적으로 지문을 입력 할 수 있습니다.

3) 지문이 뚜렷하지 않거나 상처가 있는지 확인하십시오.

- 지나치게 건조하거나 습한 지문, 흐릿한 지문, 상처가 있는 지문은 인식하기가 어렵습니다. 이런 경우에는 다른 손가락의 지문을 등록하십시오.



A.2. 잘못된 지문 입력 방법

1) 손가락을 비스듬하게 놓은 경우

- 손가락을 비스듬하게 올리면 오류 확률이 높아집니다.

2) 손가락이 입력창에 밀착이 안 됐을 경우

- 입력창에서 지문이 떨어지면 인증이 되지 않습니다.

3) 손가락 끝을 입력창에 올려놓은 경우

- 손가락 끝을 입력창에 놓으면 인증이 되지 않습니다.

4) 손가락을 돌리는 경우

- 지문 입력 시 손가락을 돌리거나 움직이면 인증률이 낮아집니다.

5) 손가락을 입력창 구성에 올려놓은 경우

- 입력창 구석에 손가락을 놓으면 인증이 되지 않습니다.

A.3. 인증 실패 시 대처 방안

1) 손가락에 물기가 많을 때

- 물기가 많으면 인증이 잘 되지 않습니다. 물기를 닦고 입력하세요.

2) 손가락이 너무 건조 할 때 (특히 겨울철)

- 손가락에 적당한 수분(입김)을 주신 후 입력하세요.

3) 손가락에 상처가 났을 때

- 다른 지문을 등록하여 사용하시는 것이 좋습니다.

4) 지문 입력창이나 손가락에 이물질이 묻었을 때

- 지문 입력창을 깨끗이 닦은 후 사용하세요.

A.4. 지문 등록, 이렇게 하면 편리합니다.

1) 지문의 상태가 선명하고 양호한 손가락을 등록해 주십시오.

2) 아동의 경우 지문의 크기가 너무 작거나 여린 경우 사용이 불편하므로, 6개월 주기로 새로 지문을 등록할 필요가 있습니다.

3) 성공률이 떨어지거나 지문의 상태가 여린 경우 같은 손가락을 2~3번 복수로 등록하면 사용이 더욱 원활해 집니다.

4) 될 수 있는 대로 검지 손가락을 이용하여 손도장을 찍는 기분으로 입력하십시오. 손가락 끝만 살짝 닿도록 입력하는 것은 올바른 입력 방법이 아닙니다. 지문의 정중앙이 지문입력창에 닿게 하십시오.

A.5. 사용자 지문 상태에 따른 유의 사항

1) 지나치게 건조하거나 습한 지문, 흐릿한 지문, 상처가 있는 지문은 인식하기가 어렵습니다. 이런 경우에는 다른 손가락의 지문을 등록하십시오.

2) 지문 입력 시에는 움직이거나 흔들리지 않도록 입력하면 사용이 더욱 원활합니다.

3) 노인의 경우 등록할 지문에 잔금이 심하면 등록이 안 될 경우가 있습니다.

4) 검지 손가락을 이용하면 정확하고 안정적으로 지문을 입력 할 수 있습니다.

5) 가급적 2개 이상의 지문을 등록해 사용하시면 좋습니다.